Targeting and Privacy in Mobile Advertising

Omid Rafieian[*] Hema Yoganarasimhan[*]
University of Washington University of Washington

April 10, 2019

[*]We are grateful to an anonymous firm for providing the data and to the UW-Foster High Performance Computing Lab for providing us with computing resources. We thank Avi Goldfarb, Daria Dzyabura, Clarence Lee, Simha Mummalaneni, Sridhar Narayanan, Amin Sayedi, K. Sudhir, and Daniel Zantedeschi for detailed comments that have improved the paper. We also thank the participants of the 2016 Invitational Choice Symposium, 2016 FTC and Marketing Science Conference, 2016 Big Data and Marketing Analytics Conference at the University of Chicago, 2017 Ph.D. Students Workshop at the University of Washington, 2017 Adobe Data Science Symposium, SICS 2017, the 2018 MSI-Wharton Conference on New Perspectives in Analytics, 2018 UW-UBC conference, the 2019 UT Dallas FORMS conference, and MIT and CMU marketing seminars for their feedback. Please address all correspondence to: rafieian@uw.edu, hemay@uw.edu.

Abstract

Mobile in-app advertising is growing in popularity. While these ads have excellent user-tracking properties through mobile device IDs, they have raised concerns among privacy advocates. This has resulted in an ongoing debate on the value of different types of targeting information, the incentives of ad-networks to engage in behavioral targeting, and the role of regulation. To answer these questions, we propose a unified modeling framework that consists of two components – a machine learning framework for targeting and an analytical auction model for examining market outcomes under counterfactual targeting regimes. We apply our framework to large-scale data from the leading in-app ad-network of an Asian country. We find that an efficient targeting policy based on our machine learning framework improves the average click-through rate by 66.80% over the current system. These gains mainly stem from behavioral information compared to contextual information. Theoretical and empirical counterfactuals show that while total surplus grows with more granular targeting, ad-network's revenues are non-monotonic, i.e., the most efficient targeting does not maximize ad-network revenues. Rather, it is maximized when the ad-network does not allow advertisers to engage in behavioral targeting. Our results suggest that ad-networks may have incentives to preserve users' privacy without external regulation.

Keywords: mobile advertising, machine learning, targeting, behavioral targeting, privacy, auctions, regulation

1 Introduction

1.1 Mobile Advertising and Targeting

Mobile advertising has grown exponentially in the last few years. Mobile advertising now constitutes the largest share of total digital ad spend (eMarketer, 2019). The rapid growth of mobile advertising partly stems from an ad format unique to the mobile environment: in-app ads (ads shown inside apps). These ads have excellent user-tracking properties, and allow ad-networks to stitch together user data across sessions, apps, and advertisers.[1] Thus one of the main attractions of in-app advertising is their ability to facilitate behavioral targeting (Edwards, 2012).[2]

While the advertising industry has lauded the trackability of in-app ads, consumers and privacy advocates have derided them citing privacy concerns. Advertisers argue that tracking allows consumers to enjoy free apps and content, and see relevant ads, whereas users demand higher privacy and limits on behavioral tracking and targeting (Edwards-Levy and Liebelson, 2017). Responding to consumer concerns, regulatory bodies have started taking action. For example, the European Union's General Data Protection Regulation agreement requires users to opt into, rather than opt out of, behavioral targeting (Kint, 2017).

Even as consumers, businesses, and regulators are trying to find the right balance between consumer protection and business interests, we do not have a good understanding of the key issues at the core of targeting and privacy. For example, to what extent does targeting improve the efficiency of the advertising eco-system, what is the value of different types of targeting information, and what are the incentives of different players in the advertising industry to engage in user-tracking and behavioral targeting? The lack of empirical analyses on these issues hampers our ability to have an informed discussion on, and to form policy, on them.

1.2 Research Agenda and Challenges

In this paper, we seek to address this gap by providing answers to the following sets of questions related to targeting and privacy in the mobile ecosystem.

The first set of questions relate to targeting and efficiency. How can ad-networks use the data available to them to develop targeting policies? How can we evaluate the performance of these policies in both factual and counterfactual settings? Specifically, what are the gains in CTR from adopting an efficient targeting policy?

[1]Advertisers and ad-networks have access to a unique device ID associated with the mobile device referred to as IDFA (ID For Advertisers) in iOS devices, and AAID (Android Advertiser ID) in Android devices. This device ID is highly persistent and remains the same unless actively re-set by the user.

[2]This is in contrast to browser-based mobile ads, which have poor tracking properties since third-party cookies do not work in many browsers (e.g., Safari) and hence tend to be poorly targeted (Sands, 2015).

The second set of questions relate to the value of targeting information. We are particularly interested in the relative value of contextual vs. behavioral information. The former quantifies the context (when and where) of an impression, and the latter constitutes historic data on an individual user's past app usage, ad exposure, and ad response. Contextual information is privacy preserving whereas behavioral information is based on user-tracking and therefore impinges on user-privacy.

Third, we are interested in quantifying the revenue-efficiency trade-off and platform's incentives to preserve user privacy. What is the empirical relationship between efficiency and platform revenues? Do ad-networks have an incentive to allow advertisers to do targeting and thereby enable targeted bidding? What is the optimal level of targeting from the perspective of different players? Finally, to what extent are the ad-network's and advertisers' incentives aligned?

There are three main challenges that we need to overcome to satisfactorily answer these questions. First, to develop efficient targeting policies, we need to obtain accurate click-through rate (CTR henceforth) or match value estimates for all counterfactual ads (i.e., ads that could have been shown), and not just the ad that was actually shown in that impression. Without sufficient randomization in the ad allocation mechanism, no model can obtain accurate match value estimates for the entire set of competing ads. Thus, we need a setting with sufficient randomness in the ad allocation mechanism to develop counterfactual targeting policies. Second, to correctly quantify the value of different pieces of targeting information, we need a model that can accurately predict whether a targeted impression will lead to a click or not. Models with poor predictive ability will lead to downward bias and noise in the estimates of the value of information. Third, we need an underlying model of strategic interactions that can quantify market outcomes such as surplus and revenue under different targeting levels offered by the ad-network. Without an economic model of market players that puts some structure on revenues, we cannot make any statements on the ad-network's or advertisers' incentives to target and/or the extent to which these incentives are aligned.

1.3 Our Approach

We present a unified and scalable modeling framework that coherently combines predictive machine learning models with prescriptive economic models to overcome the challenges listed above. Our framework consists of two main components. The first, a machine learning framework for targeting, addresses the first two challenges of obtaining counterfactual match value estimates and achieving high predictive accuracy. The second, an analytical model that incorporates competition and characterizes the ad-network's and advertisers' profits under different targeting regimes. This addresses the third challenge of linking targeting strategies to revenues.

The machine learning module consists of three parts – (a) a filtering procedure, (b) a feature generation framework, and (c) a learning algorithm. The filtering procedure takes any impression and determines the set of ads that have a non-zero probability of being shown in that impression, i.e., counterfactual ads for which we can obtain accurate CTR estimates. Clearly, if the ad allocation mechanism selects ads determinstically for each impression, we are only able to generate accurate estimates for the ad that is shown in that impression. While deterministic auctions are the common practice in this industry, we take the advantage of our unique data setting that utilizes a probabilistic allocation mechanism, which provides sufficient randomization in the data generating process. Our feature generation framework relies on a set of functions to generate a large number of features that capture the contextual and behavioral information associated with an impression. Using these functions, we generate a total of 160 features that serve as input variables into a CTR prediction model. Finally, we use XGBoost, proposed by Chen and Guestrin (2016), a fast and scalable version of boosted regression trees, as our learning algorithm.

In the second component of our model, we address the question of ad-network's incentives to allow targeting. In an influential paper, Levin and Milgrom (2010) conjecture that high levels of targeting can soften competition between advertisers and thereby reduce ad-network's revenues. To examine this idea, we develop an analytical framework that incorporates a model of auction with targeting where advertisers can place targeted bids given the targeting level they are provided with. Our framework allows us to characterize the the ad-network's revenue, advertisers' surplus, and total surplus under different targeting regimes both theoretically and empirically. While the extent to which we can analyze this problem theoretically is limited without further assumptions on distribution of valuations, we can use it to characterize ad-network revenues and efficiency for any targeting level if we have access to empirical estimates for advertisers' valuations per impression. To do so, we need to estimate two sets of primitives – (a) advertisers' click valuations, (b) match valuations or CTRs. The former can be estimated from the observed bids under the current auction used by the ad-network. For the latter, we use our machine learning framework for targeting that gives us click probabilities for all competing ads for any impression. The product of these two then gives us value per impression for any advertiser, thereby allowing us to empirically evaluate market outcomes under any targeting regime.

We apply our framework to one month of data from the leading mobile advertising ad-network from a large Asian country. The scale and scope of the data are large enough to provide realistic substantive and counterfactual results. For our analysis, we sample over 27 million impressions for training, validation, and testing, and use another 146 million impressions for feature generation. A notable feature of our data is the extent of randomization of ads over impressions – ads are shown over a large variety of impressions and impressions can be allocated to a wide variety of ads, i.e., the targeting/match between ads and impressions is quite low. A few unique aspects of our marketplace contribute to this feature, such as the use of a quasi-proportional auction and the low levels of targeting in the current system. This randomization greatly enables our analysis, especially when it comes to counterfactuals.

1.4 Findings and Contribution

We first discuss the results from the machine learning model for targeting. We present both factual and counterfactual evaluations of our model. In the factual evaluation, we use goodness-of-fit measures to evaluate how well our model can predict the observed outcome. We find that our framework predicts the outcome on a hold-out test set with substantial accuracy: the Relative Information Gain (*RIG*) is 17.95% for our model over the baseline that simply predicts the average CTR for all impressions. We then pin down the role of different pieces of information and show that behavioral information is more useful that contextual information in predicting CTR: the *RIG* for the behavioral model (without using any contextual information) is 12.14%, while it is 5.25% for the contextual model (without using any behavioral information). Thus, from an efficiency perspective, behavioral targeting is more effective than contextual targeting. In the second part of our evaluation, we consider the CTR-maximizing counterfactual assignment wherein each impression is allocated to the ad with the highest estimated CTR in that impression. We show that this efficient targeting policy can increase the average CTR in the ad-network by 66.80% over the current system.

We then address the question of ad-network's incentives to allow targeting. First, we theoretically prove that – (a) the total surplus in the system monotonically increases as the extent of targeting increases, but (b) the ad-network's revenues are not monotonic; it may or may not increase with more granular targeting. As a special case, we then consider four targeting regimes that relate to our research question – full, behavioral, contextual, and no targeting. We prove that full targeting maximizes surplus while no targeting minimizes surplus. Behavioral and contextual targeting can both be interpreted as partial targeting and lie in between. However, because we cannot characterize one as more granular than another, theory does not offer us any guidance here. Thus, we need an empirical analysis to quantify the relative magnitude of surplus and revenues under these two cases.

Our empirical analysis reveals that ad-network revenues are maximized when it restricts targeting to the contextual level even though doing so lowers total surplus, i.e., allowing behavioral targeting thins out the market, which in turn reduces ad-network revenues. Therefore, the ad-network may be incentivized to adopt a privacy-preserving targeting regime, especially if it cannot extract additional surplus from advertisers through other mechanisms. On the advertisers' side, although a majority of them prefer a regime where the ad-network allows behavioral targeting, not all do. An important implication of our findings is that it may not be necessary for an external entity such as EU/FCC to impose privacy regulations.

In sum, our paper makes several contributions to the literature. First, from a methodological perspective, we propose a novel machine learning framework for targeting that is compatible with counterfactual analysis in a competitive environment. A key contribution of our machine learning framework is in combining existing ideas from causal inference literature (e.g., non-zero propensity score) with recent machine learning literature to generate counterfactual estimates. Further, we present an auction model with targeting that characterizes advertisers' utility function for any targeting regime and provides a direct link to the estimation of market outcome such as efficiency and revenue. Second, from a substantive perspective, we provide a comprehensive comparison of contextual and behavioral targeting, with and without the presence of competition. To our knowledge, this is the first study to compare the role of behavioral and contextual targeting on market outcomes. Third, from a managerial perspective, our results demonstrate a non-monotonic relationship between targeting granularity and revenues. While our findings may depend on the context of our study, our framework is generalizable and can be applied to any auction setting that induces some randomization in the allocation of ads. Finally, from a policy perspective, we identify the misalignment of the ad-network's and advertisers' incentives regarding behavioral and contextual targeting and information disclosure. We expect our findings to be of relevance to policy-makers interested in regulating user-tracking and behavioral targeting in the mobile advertising space.

2 Related literature

First, our paper relates to the computer science literature on CTR prediction (McMahan et al., 2013; He et al., 2014; Chapelle et al., 2015). These papers make prescriptive suggestions on feature generation, model selection, learning rates, and scalability. Although we build our machine learning framework following their approaches, our work differs in two main ways. First, we develop a filtering procedure that allows us to obtain accurate CTR estimates for both the ad shown during an impression as well as counterfactual ads not shown in the impression. Thus, unlike the previous papers, our framework can be used to develop and evaluate different targeting policies. Second, we quantify the value of different types of information in the targeting of mobile in-app ads, whereas the previous papers were mainly concerned with presenting scalable methods for predicting clicks. Our paper also relates to the growing literature on applications of machine learning in marketing (Toubia et al., 2007; Dzyabura and Yoganarasimhan, 2018).

Our paper also relates to the literature on ad-network's incentives to allow targeting. Levin and Milgrom (2010) were one of the first to conjecture the trade-off between value creation and market thickness. They argue that too much targeting can thin markets, which in turn can soften competition and make the ad-network worse off. This is particularly the case when there is significant heterogeneity in the distribution of advertisers' valuation of impressions (Celis et al., 2014). Building on this insight, a growing stream of analytical papers show that there is a non-monotonic pattern between the extent of targeting and ad-network revenues (Bergemann and Bonatti, 2011; Fu et al., 2012; Amaldoss et al., 2015; Hummel and McAfee, 2016; De Corniere and De Nijs, 2016; Sayedi, 2018). A key difference between these papers and ours is that we do not make any distributional assumptions on the match values in our analytical formulation.

In spite of the increasing interest from the theoretical side, there has been limited empirical work on this topic with mixed findings. In an early paper, Yao and Mela (2011) present a structural model to estimate advertisers' valuations and show that targeting benefits both advertisers and the ad-network. In a similar context, however, Athey and Nekipelov (2010) present a case study of two keywords and show that coarsening CTR predictions (worse targeting) can help a search advertising ad-network generate more revenue. Our paper speaks to this debate by presenting a large-scale study that establishes that too much behavioral targeting can adversely impact ad-network revenues. More broadly, neither of these papers are able to compare the ad-network's incentives to allow behavioral vs. contextual targeting because: (a) they do not have sufficient randomization in the data generating process and (b) they do not build a targeting framework that can be used to develop different targeting policies and evaluate their efficiency. To our knowledge, our paper is the first empirical paper to view revenue-efficiency trade-off through the lens of privacy and examine whether the platform has natural incentives to preserve users' privacy.

Finally, our paper relates to the literature on the interplay between privacy and targeting. Goldfarb and Tucker (2011b) use data from a series of regime changes in advertising regulations and show that restricting targeting reduces response rates and thereby advertisers' revenues. Similarly, Goldfarb and Tucker (2011a) and Tucker (2014) highlight the perils of excessive targeting as users perceive increased targeting as a threat to their privacy. Using a structural approach, Johnson (2013) finds that both advertisers and publishers are worse off when the ad-network introduces stricter privacy policies that reduce targeting. Please see Goldfarb (2014) for an excellent review of targeting in online advertising and Acquisti et al. (2016) for a detailed discussion of consumer privacy issues. Thus, our paper is the first work to provide empirical evidence for the possibility of self-regulation in this market.

3 Setting and Data

3.1 Setting

Our data come from the leading mobile in-app advertising network of a large Asian country, which had over 85% market-share in the category in 2015. The ad-network works with over 10,000 apps and 250 advertisers and it serves over 50 million ads per day (about 600 auctions per second).

3.1.1 Players

There are four key players in this marketplace.
Users: Individuals who use apps. They see the ads shown within the apps that they use and may choose to click on the ads.
Advertisers: Firms that show ads through the ad-network. They design banner ads and specify their bid as the amount they are willing to pay per click, and can include a maximum budget if they want to. Advertisers can target their ads based on the following variables: app category, province, connectivity type, time of the day, mobile operators, and mobile brand of the impression. The ad-network does not support more detailed targeting (e.g., behavioral targeting) at this point in time.
Publishers: App developers who have joined the ad network. They accrue revenues based on the clicks generated within their app. Publishers earn 70% of the cost of each click in their app (paid by the advertiser), and the remaining 30% is the ad-network's commission.
Ad-network or Platform: functions as the matchmaker between users, advertisers, and publishers. It runs a real-time auction for each impression generated by the participating apps and shows the winning ad during the impression. The platform uses a CPC pricing mechanism, and therefore generates revenues only when clicks occur. [3]

3.1.2 Auction Mechanism

The platform uses a *quasi-proportional* auction mechanism (Mirrokni et al., 2010). Unlike other commonly-used auctions (e.g., second price or Vickrey), this auction uses a probabilistic allocation rule:

$$\pi_{ia} = \frac{b_a q_a}{\sum_{j \in \mathcal{A}_i} b_j q_j} \tag{1}$$

where π_{ia} is the probability that advertiser a with bid b_a and quality score q_a wins impression i, and $\mathcal{A}_i$ denotes the set of advertisers participating in the auction for impression i. The quality score is an aggregate measure that reflects the advertiser's potential profitability for the platform. Currently, the platform does not use impression-specific quality scores; rather it uses an advertiser-specific quality score that remained constant during our observation period.

Because of the probabilistic nature of the auction, the ad that can generate the highest expected revenue for the platform is not guaranteed to win. Rather, advertiser a's probability of winning is proportional to $b_a q_a$.[4]

[3]An impression lasts one minute. If a user continues using the app beyond one minute, it is treated as a new impression and the platform runs a new auction to determine the next ad to show the user.

[4]From a practical perspective, probabilistic auctions ensure that individual users are not exposed to the same ad repeatedly within the same app-session (which can be irritating). In contrast, in a deterministic auction, the same advertiser would win all the impressions unless his budget runs out.

3.2 Data

We have data on all the impressions and corresponding clicks (if any) in the platform for a 30-day period from 30 September, 2015, to 30 October, 2015. For each impression, we have data on:

- Time and date: The time-stamp of the impression.
- AAID: Android Advertising ID is a user re-settable, unique, device ID that is provided by the Android operating system.[5] It is accessible to advertisers and ad networks for tracking and targeting purposes. We use it as the user-identifier in our main analysis.
- App ID: A unique identifier for apps that advertise through the platform.
- Ad ID: Identifier for ads shown in the platform.
- Bid: The bid that the advertiser has submitted for her ad.
- Location: This includes the province as well as the exact location of a user based on latitude and longitude.
- Connectivity type: It refers to the user's type of connectivity (e.g., Wi-Fi or cellular data).
- Smartphone brand: The brand of user's smartphone (e.g., Samsung, Huawei, etc.).
- MSP: The user's mobile-phone service provider.
- ISP: The user's Internet service provider.
- Click indicator: This variable indicates whether the user has clicked on the ad or not.

The total data we see in this one month interval is quite large. Overall, we observe a total of 1,594,831,699 impressions and 14,373,293 clicks in this time-frame, implying a 0.90% CTR. Notably, we have access to all variables that the ad-network has. Thus, our analysis fully mimics the ad-network's problem.

3.3 Data Splits and Sampling

We have a snapshot of one month of data, from September 30 to October 30. We use penultimate two days (October 28 and 29) for training and validation, and the last day for testing (October 30). We also use the preceding history from September 30 to Oct 27 (referred to as global data) to generate the features associated with these impressions. The splits of data are shown in Figure 1. Note that we do not fit our model on the global data because we do not have sufficient history to generate features for these impressions. Further, constraining all the three data-sets – training, validation, and testing – to a three-day window has advantages because recent research has shown that data freshness plays an important role in CTR prediction, i.e., using older history for prediction

[5]Apple's app store is not available in the country where our data are sourced from. Hence, all smartphones use the Android operating system.

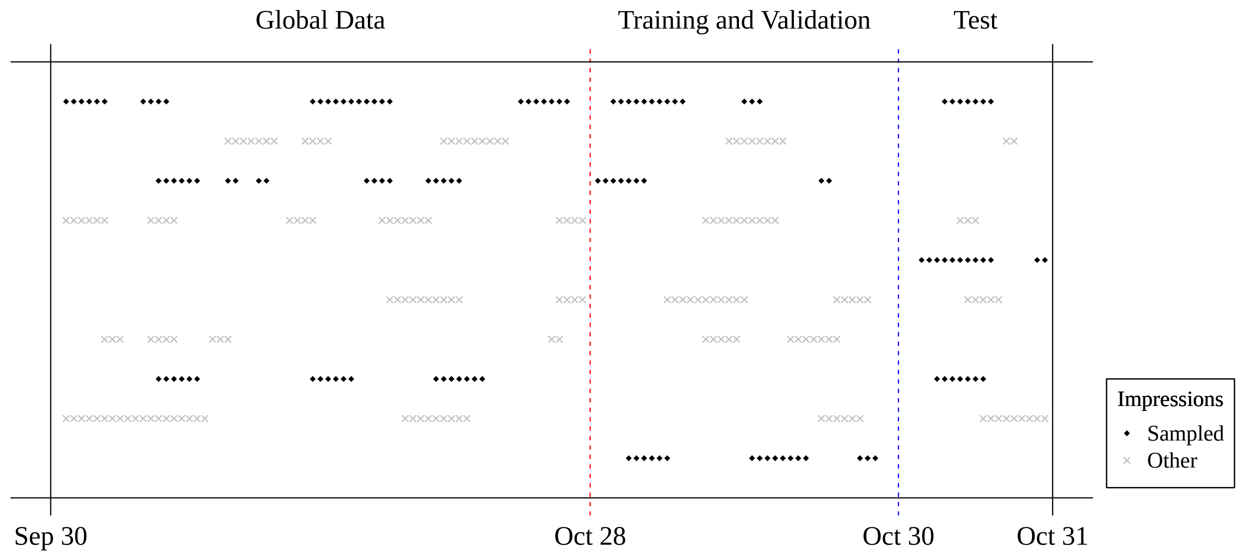


Figure 1: Schema for data generation.

can lead to poor predictive performance (He et al., 2014).[6]

We draw a sample of 728,340 unique users (out of around 5 million) seen on October 28, 29, and 30 to form our training, validation, and test data-sets.[7] In Appendix D.4, we show that this sample size is sufficient and that larger samples do not significantly improve model performance.

Figure 1 presents a visual depiction of the sampling procedure. Rows represent users. The impressions by users in our sample are shown using black points. There are 17,856,610 impressions in the training and validation data, and 9,625,835 impressions in the test data. We have an additional 146,825,916 impressions by these users in the time preceding October 28, which form global data. These impressions will be solely used for feature generation (and not for model fitting). Note that both our user-based sampling procedure and feature generation approach (see §B) require us to be able to identify and track users. For this purpose, we use the AAID variable as our user identifier.

3.4 Summary Statistics

We now present some summary statistics on our training, validation, and test data, which constitutes a total of $27,482,444$ impressions.

Table 1 shows the summary statistics of the categorical variables in the data. For each variable,

[6]As a robustness check, we consider alternative data splits and verify that, by and large, the performance of the model does not change.

[7]Another approach would be to randomly sample impressions in each split of the data. However, this would not give us the complete user-history for each impression in the training, validation, and test data-sets. This in turn would lead to significant loss of accuracy in user-level features, especially since user history is sparse. In contrast, our user-based sampling approach gives us unbroken user-history.

Variable	**Number of categories**	**Share of top categories** 1st	2nd	3rd	**Number of impressions**
App	9709	37.12%	13.56%	3.05%	27,482,444
Ad	263	18.89%	6.71%	6.31%	27,482,444
Hour of the Day	24	7.39%	7.32%	6.90%	27,482,444
Province	31	25.25%	6.65%	6.51%	21,567,898
Smartphone Brand	8	46.94%	32.30%	9.53%	25,270,463
Connectivity Type	2	54.64%	45.36%		27,482,444
ISP	9	68.03%	14.02%	7.09%	10,701,303
MSP	3	48.57%	43.67%	7.76%	26,051,042

Table 1: Summary statistics for the categorical variables.

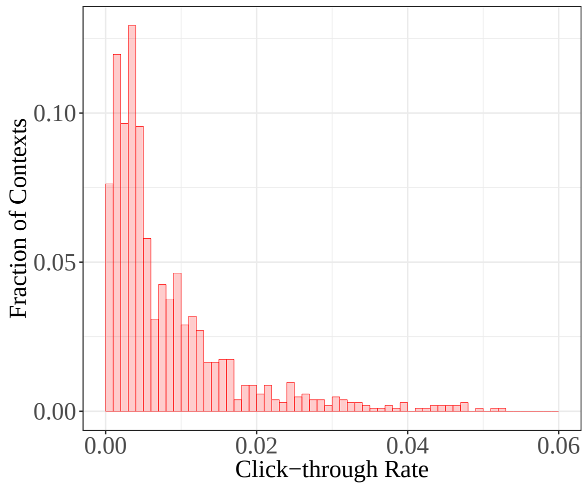


Figure 2: Histogram of click-through rates (CTR) for different contexts. Context is defined as a unique combination of an app and an hour of the day (where and when of an impression)

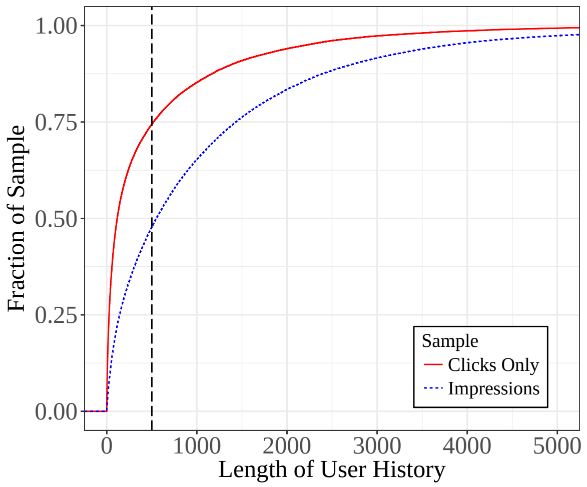


Figure 3: Empirical CDF of the length of user history for impressions and clicks (truncated at 5000). History is defined as the number of previous impressions from September 30 till the current impression.

we present the number of unique values, the share of top three values that the categorical variable can take, and the number of non-missing data. While we always have information on the app, ad, and time-stamp of the impression, the other variables are sometimes missing. The shares are shown after excluding the missing variables in the respective category.

We observe a total of 263 unique ads and 9709 unique apps in the data. The top three subcategories in each have large shares and there is a long tail of smaller apps and ads. Moreover, we find that the top 37 ads account for over 80% of the impressions, and similarly, the top 50 apps account for 80% of impressions.

Next, we present some descriptive analysis that examines the role of contextual and behavioral information in predicting CTR. A context is characterized by the 'when' and 'where' of an impression. As such, we define a unique context as a combination of an app and a specific hour of the day. Figure 2 shows the histogram of CTR for different contexts. As we can see, there is a significant amount of variation in CTR across contexts, which suggests that contextual information can be informative for predicting clicks. Next, to understand the role of behavioral information, we focus on the length of history available for a user. Figure 3 shows the CDF of the length of history for all the impressions and clicks. It suggests that users with longer histories are less sensitive to ads. Most of the clicks come from users with shorter histories, while most impressions come from users with longer histories. Thus, user-history or behavioral information also seem helpful in explaining the clicking behavior observed in data.

4 Machine Learning Framework for Targeting

In this module, our goal is to develop a framework that can accurately estimate the gains in efficiency or the CTR for any targeting policy. To do that, we first need to specify and train a machine learning model that accurately predict the match between an impression and an ad, i.e., predict whether an impression will generate a click or not, for both factual and counterfactual ads.

This section is organized as follows. We first define our problem in §4.1. Next, in §4.2, we discuss our empirical strategy. Here, we explain the need for, and the extent of, randomization in our data generating process, and propose a filtering approach that establishes the scope of our framework in estimating both factual and counterfactual targeting policies. In §4.3, we present the details of our feature generation framework. Finally, in §4.4, we discuss our estimation procedure which consists of the learning algorithm, the loss function, and the validation method.

4.1 Problem Definition

Consider a setting with N impressions and A ads. We begin with a formal definition of a targeting policy.

Definition 1. A targeting policy τ is defined as a mapping between impressions to ads such that each impression is allocated one ad. For example, $\tau(i) = a$ means that targeting policy τ selects ad a to be shown in impression i.

In order to evaluate the effectiveness of a targeting policy, we first need an accurate prediction of CTR for each ad for a given impression in our data. That is, for each impression i and ad a, we need to estimate $\Pr(y_{i,a} = 1)$, where $y_{i,a}$ is the indicator that ad a receives a click when it is shown in impression i. This brings us to the formal definition of match value matrix:

Definition 2. Let $m_{i,a} = \Pr(y_{i,a} = 1)$. The $N \times A$ match value matrix M is defined as:

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \dots & m_{1,A} \\ m_{2,1} & m_{2,2} & \dots & m_{2,A} \\ \vdots & \vdots & \ddots & \vdots \\ m_{N,1} & m_{N,2} & \dots & m_{N,A} \end{bmatrix}, \tag{2}$$

where N denotes the total number of impressions in our data and A denotes the total number of ads competing for these impressions.

In this section, our goal is to develop a machine learning framework to estimate this match value matrix. We can use our estimated match value matrix, $\hat{M}$, to perform the following analyses:

1. **Evaluate model performance:** We can evaluate the predictive performance of our model using the observed outcome. Let τ_0 denote the *current targeting policy*, such that:

$$\tau_0(i) = a_i, \tag{3}$$

   where a_i is the ad that is actually shown in impression i. Since we observe the actual outcomes for y_{i,a_i}, we can evaluate how well our $\hat{m}_{i,a_i}$'s estimate these outcomes.

2. **Evaluate the gains from efficient targeting policy:** Using the match value matrix, we can evaluate the expected CTR of any counterfactual targeting policy τ as follows:

$$\hat{m}^{\tau} = \frac{1}{N} \sum_{i=1}^{N} \hat{m}_{i,\tau(i)} \tag{4}$$

   In particular, we are interested in the *efficient targeting policy*, τ^*, determined by our model that allocates each impression to the ad with the highest CTR for that impression:

$$\tau^*(i) = \operatorname*{argmax}_a \hat{m}_{i,a} \tag{5}$$

   In §5.2, we quantify the gains in average CTR from efficient targeting over the current system.

4.2 Empirical Strategy

We now present our empirical strategy to estimate matrix M. At a high-level, our goal is to build a model to predict whether an impression i showing ad a will receive a click or not, based on the joint distribution of impressions and clicks in our data. That is, we seek to estimate a function $f(X_{i,a})$

such that:

$$m_{ia} = Pr(y_{i,a} = 1) = f(X_{i,a}), \tag{6}$$

where $X_{i,a}$ is a set of features that are informative of whether impression i showing ad a will receive a click. Since this problem can be interpreted as function evaluation, we turn to machine learning algorithms that can capture complex relationships between the covariates and the outcome without imposing strong parametric restrictions on $f(\cdot)$.

The estimates from our machine learning approach are accurate as long as the joint distribution of covariates and outcome (click) in the training set is the same as that in the test set. In other words, we require the training and test sets to have the same data generating process. While this assumption is satisfied for the set of ads that are actually shown in the data, this may not hold for ads that are not shown. As such, our machine learning model is only guaranteed to provide one accurate estimate per row – for the ad that is actually shown. For the rest of ads, the estimates are accurate only if they *could* have been shown in that impression, i.e., have a non-zero probability of being shown in that impression given the data generating process. Thus, for each impression, our estimates are accurate for the set of counterfactual ads that could have been shown in that impression. This is because, in a sufficiently large data-set, this set of ads would have been shown on impressions similar to the focal impression, which allows us to estimate the match value for all these impression-ad combinations.

It is worth noting that if the platform runs a deterministic auction (e.g., second price auction), the set of ads that could have won the auction (and hence shown during an impression) is a singleton. Similarly, the set of ads that can be shown in an impression in highly targeted environments would be very small. Therefore, data-sets generated without any randomization in the ad allocation mechanism will not allow researchers to push the scope of their analysis beyond the set of actual outcomes observed in the data. Randomization in ad allocation is thus necessary if we want to use our framework to evaluate the effectiveness of counterfactual targeting policies. In §4.2.1, we provide a discussion on the need for and the extent of randomization in our data. Next, in §4.2.2, we propose a filtering procedure to identify the set of ads for which our estimates are accurate.

4.2.1 Randomization and Accuracy of Estimates

We now discuss why we need randomization in the ad allocation mechanism and the extent of randomization necessary. We then discuss the sources of randomization in our data, and present some empirical evidence on the extent of randomization.

Why do we need randomization?

Randomization in the data generating process is not necessary to build a good predictive model of clicks. In other words, if we do not use the model to predict in counterfactual situations that are "not seen in the data", the lack of randomization is not a problem, because the training and test sets have the same joint distribution of covariates and clicks. Thus, randomization is not necessary if we only want to build a model to predict the CTR of the ads shown in the data (i.e., the factual element for each row of the matrix M), and use it to evaluate the contribution of each piece of information in predicting the observed outcome.

However, since we are also interested in evaluating the effectiveness of counterfactual targeting policies, we need accurate estimate all the elements of the matrix M. To get these estimates, we need randomization in ad allocation. Intuitively, our predictions of CTRs for the competing ads for any impression will be inaccurate if these ads would never have been shown in that impression. For example, if an ad is only shown in App X in our data, then no model (ML or otherwise) will be able to accurately predict the ad's CTR in App Y. Therefore, we need to see sufficient randomness in the allocation of ads across impressions to ensure that the joint distribution of covariates and clicks in our counterfactuals is the same as that in training data.

To what extent do we need randomization?

While the ideal case is a fully randomized experiment, we argue that if a counterfactual ad has a non-zero probability of being shown in an impression, then we can recover an accurate estimate of its CTR using a large-scale training set. Naturally, estimates for small ads with a very small probabilities of winning will be noisy. However, it is possible to overcome this issue by focusing on the top 37 ads that constitute over 80% of our data. In §4.2.2, we describe how we filter ads to ensure our estimates are accurate for the counterfactual ads.

What are the sources of randomization in our data generating process?

There are three main factors that contribute to the randomness in our data. First, it is the probabilistic nature of our auction. Unlike deterministic auctions where the highest scoring bidder always wins, in quasi-proportional auctions all bidders have a chance of winning. Therefore, ads end up being shown over a wide range of impressions. Second, the quality score q_a used in the auction for ad a is constant for the ad across all impressions and is not regularly updated. Thus, the platform does not do really match ads to impressions. Third, there are only few categories over which advertisers can target their ads and the extent of targeting happening in the system is low. Table 2 shows the list of variables over which advertisers can now target and the percentage of top advertisers that target on each of these variables. Interestingly, more than half of the top ads do not target in any given targeting area and even those that do target, do so minimally. Moreover, the platform does not allow any behavioral or user-level targeting. Because of these reasons, the extent of randomization of ads over impressions is quite high.

Targeting Area	No. of Top Ads Targeting
App Category	13
Province	12
Connectivity Type	2
Time of the Day	9
MSP	1
Mobile Brand	1
ISP	1

Table 2: Targeting decisions of the top 37 ads.

What is the extent of randomness in our data?

We now provide some empirical evidence to illustrate the extent of randomness in our data. A unique aspect of our ad placement mechanism is that ads are changed and replaced sequentially within a session. After an ad has been shown for one minute within a session, it is replaced by an ad chosen through the same auction mechanism as the previous ad. In deterministic auctions, the same ad will be shown repeatedly since bids and CTRs are not updated within a session. However, since our auction mechanism is probabilistic, a user can be exposed to multiple ads within the same session even though an identical auction runs for all impressions within the session.

To illustrate this point, we calculate the number of distinct ads shown in a session prior to a given exposure/impression number. We then plot the empirical CDF of the number of distinct ads shown within a session at the sixth, eighth, and tenth exposures in Figure 4. As we can see, there is substantial variation in the number of distinct ads at each exposure number. For example, at least three of the first five ads are distinct in 80% of the sessions. This implies that the extent of randomization in our data is quite high.

4.2.2 Filtering Procedure

While randomization helps us obtain accurate CTR estimates for a wide range of counterfactual ads, there remains a set of ads for which our estimates will not be accurate. Recall that because our auction mechanism is probabilistic, all ads participating in the auction for an impression i have a non-zero probability of winning. Thus, the only ads that have a zero probability of being shown in an impression are ones that are not available to participate in the auction for it. Our goal is to identify and filter such ads for each impression i in our data. Let $E_{N\times A} = [e_{i,a}]$ denote a filtering matrix, where each element $e_{i,a}$ takes value 1 if ad a has a non-zero probability of winning impression i, and 0 otherwise. We then construct each row i of $E_{N\times A}$ by identifying ads that were not available to participate in the auction for impression i. Two factors influence whether an ad is available to participate in an auction for an impression:

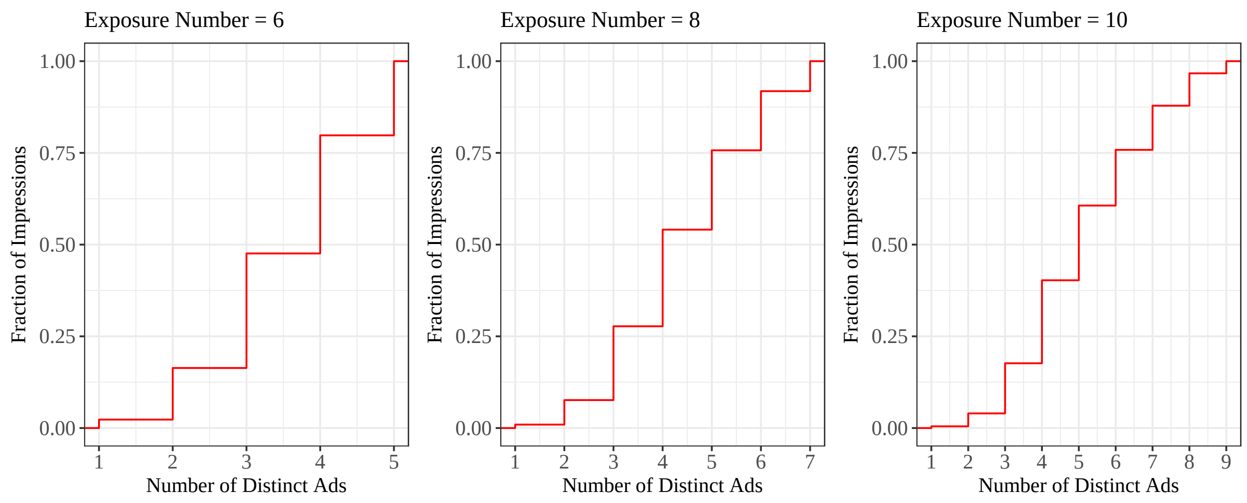


Figure 4: Empirical CDF of the number of distinct ads for three exposure numbers: 6, 8, and 10, in the training and test data.

- Targeting: Targeting by advertisers is the main reason why some ads are unavailable to compete for certain impressions, and therefore have zero probability of being shown in them. For example, if an ad chooses to target only mornings, then it is not considered in the auctions for impressions in evenings. In that case, we should filter out this ad for all impressions in the evening. While limited, targeting is nevertheless present in our setting, and mainly happens on province, time, and app categories (see Table 2). Hence, for each impression i, we filter out all ads that were excluded from the auction for i because of targeting.
- Campaign availability: Second, some ads may be unavailable to compete for a given impression because their ad campaigns may not be running in the system when the impression happens. This could happen either because the advertiser's budget has been exhausted, or because the advertiser has exited the market. Therefore, for each impression i, we filter out ads that were unavailable when it happens. Empirically, we find that campaign availability is not a major factor that leads to ad-filtering since we focus on top ads.[8]

We construct the $E_{N \times A}$ matrix by filtering out ads for each impression based on the factors discussed above. Each row in this matrix shows which ads are competing for an impression. However, our filtering may not be accurate for observations with missing targeting variables. Therefore, for all the analyses that use filtering, we only focus on the *Filtered Sample*, which consists of the impressions in the test data for which all targeting variables are non-missing. Figure 5 shows the empirical CDF of the number of competing ads for each impression in the Filtered

[8]Only six ads experience budget exhaustion (at least once) in the training data, four of which are completely out for auctions in the test data.

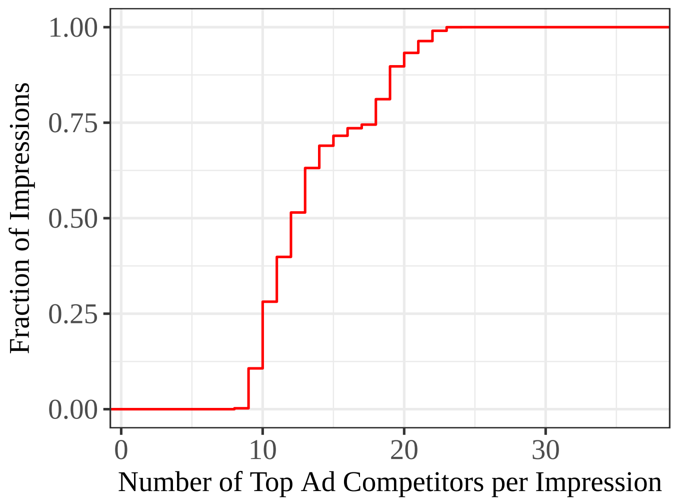


Figure 5: Empirical CDF of the number of competitors (from the top 37 advertisers) per impression on the Filtered Sample.

Sample data among top 37 ads per impression. Note that almost all impressions have at least 8 ads competing for it, and the median impression has 13 competitors. This pattern, together with the probabilistic auction mechanism, ensures that there is sufficient randomization in ads across impressions (as shown in Figure 4).

4.3 Feature Generation Framework

As discussed in §4.2, our goal is to build a model, $m_{ia} = Pr(y_{ia} = 1) = f(X_{ia})$ to accurately predict whether an impression i will receive a click or not. As such, we first need a vector of features $X_{i,a}$ that captures the factors that affect whether the user generating impression i will click on ad a.

It is important to generate an exhaustive and informative set of features since the predictive the accuracy of our model will largely depend on the quality of features we use. Given our research agenda, our features should also be able to capture the contextual and behavioral information associated with an impression over different lengths of history preceding the impression (long-term, short-term, and session-level). To achieve these objectives, we adopt the main ideas from the functional feature generation framework proposed by Yoganarasimhan (2019). There are three advantages of doing so. First, her function-based approach allows us to generate a large and varied set of features using a parsimonious set of functions. Second, it allows for a natural mapping between feature inputs and feature classification. Third, the general class of features she suggests have been shown to have good predictive power in this class of problems.

We now present a short overview of our feature functions and feature categorization below, and refer interested readers to Appendix §B for a more detailed description.

4.3.1 Inputs for Feature Functions

To generate a set of features for each impression, we use feature functions that take some inputs at the impression level and output a corresponding feature for that impression. Our feature functions typically need two types of inputs:

- *Impression-specific information:* Each impression in our data can be uniquely characterized by three types of information – (1) contextual information that captures the context (where and when) of the impression, i.e., which app serves this impression and at what time (hour of day) is the impression being shown, (2) behavioral information that denotes the identity of the user generating this impression, and (3) ad-related information, that denotes the identity of the ad which was shown during this impression.
- *History:* This input characterizes the history over which we aggregate to calculate the output of our functions. We define three different levels that capture the long-term (approximately one month), short-term (3 days), and ongoing session-level history. Besides, we characterize the history in such a way that we can update the features in real-time.

To reduce the dimensionality of our feature sets and boost the speed of our feature generation framework, we group the smaller apps (below top 50) into one app-category and all the smaller ads (below top 37) into one ad-category. Thus, our features do not distinguish the context of smaller apps (ads) as separate from each other, though they are able to distinguish them from the top apps (ads). Please see Appendix §B.1 for a complete formal definition of the inputs for feature functions.

4.3.2 Feature Functions

One challenge we face is that most of the information characterizing an impression-ad combination is categorical in nature, e.g., the app showing the ad, the user seeing the ad. As a result, approaches that include all these categorical raw inputs and their interactions as covariates are prone to the curse of dimensionality. So we define functions that take these raw inputs as well as their interactions and map them onto a parsimonious set of features that reflect the outcome of interest – CTR.

We present an overview of our feature functions in Table 3 along with their functionality (see Appendix §B.2 for a detailed description of the feature functions). These functions take different inputs based on the focal impression and return outputs that are integers or real numbers. These inputs are basically interactions of different raw inputs. The following examples give a high-level overview of what these functions do. Let p_i, t_i, u_i, and a_i denote the app, hour, user, and ad associated with impression i. If the function *Impressions* is given p_i, u_i, and a_i and long-term history as inputs, it simply returns the number of times user u_i has seen ad a_i inside app p_i from the start of the data till the time at which impression i occurred. However, if it is only given u_i and

Function	Functionality
Impressions	Number of impressions for a given set of inputs over a pre-specified history
Clicks	Number of clicks for a given set of inputs over a pre-specified history
CTR	Click-through rate for a given set of inputs over a pre-specified history
AdCount	Number of distinct ads shown for a given set of inputs over a pre-specified history
Entropy	Dispersion of ads shown for a given set of inputs over a pre-specified history
AppCount	Number of distinct apps used by a given set of inputs over a pre-specified history
TimeVariability	Variance in the user's CTR at different hours of the day over a pre-specified history
AppVariability	Variance in the user's CTR across different apps over a pre-specified history

Table 3: Feature functions

short-term history, it returns number of impressions user u_i has seen across all apps and ads over the last three days. Using this logic, we give different sets of inputs to these functions and generate 98 features for each impression i. In addition, we include a few standalone features such as dummies for each of the top ads, the user's mobile and Internet service providers, latitude, longitude, and connectivity type. Overall, we have a total of 160 features for each impression-ad (ia) combination. Together, these features form the set of predictive covariates, X_{ia}, for the impression i showing ad a. for Please see Appendix §B.3 for the full list of features.

4.3.3 Feature Categorization

All our features capture one or more type of information – contextual, behavioral, and ad-specific. To aid our analysis, we therefore classify features based on the type of information used to generate them and group them into the following (partially overlapping) categories:

- *Contextual* features (F_C): These are features that contain information on the context of the impression – app and/or hour of the day.
- *Behavioral* features (F_B): These are features that contain information on the behavior of the user who generated the impression.
- *Ad*-specific features (F_A): These are features that contain information on the ad shown during the impression.

The three feature sets form our full set of features $F_F = F_B \cup F_C \cup F_A$. We now present a few examples of features generated using the $Clicks$ function to elucidate this classification. The total clicks made by user u_i across all apps, ads, and hours of the day in the past month is a purely behavioral feature since it only contains information on the behavior of the user who generated impression i. On the other hand, the total clicks made by user u_i in the app p_i over the last month is both a behavioral and contextual feature since it contains information on both the behavior of u_i as well as the context (app p_i) in which she made these clicks. Finally, the total clicks received by ad a_i over the last one month across all users, apps, and times is a purely ad-specific feature since

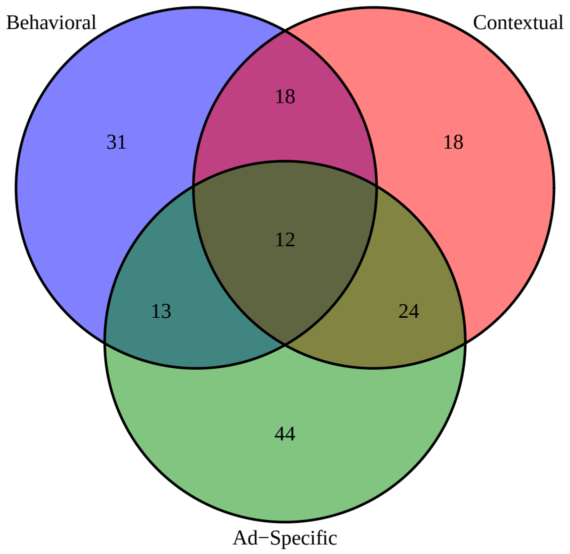


Figure 6: Venn diagram of the three feature sets, with the number of features in each region.

it only reveals information about the ad's propensity to receive clicks. Thus a feature can contain any combination of behavioral, contextual, or ad-specific information depending on the inputs used to generate it. Please see Table A1 in the Appendix for a mapping between each feature and the categories that it falls under and Figure 6 for a Venn diagram of our classification system.

4.4 Learning Algorithm: XGBoost

We now discuss the final step of our machine learning framework – the learning algorithm, which helps us learn the function $f(X_{i,a})$. It provides a mapping between our feature set $(X_{i,a})$ and the match value or click probability, as: $f(X_{i,a}) = m_{i,a} = Pr(y_{i,a} = 1)$. Given that we want to maximize the predictive accuracy of the model, we do not want to impose parametric assumptions $f(\cdot)$. The problem of function evaluation is fundamentally different and harder than the standard approach used in the marketing literature, wherein we simply evaluate parameters after assuming a functional form. In the latter, the researcher only needs to search over the set of parameters given functional form, whereas in the former we have to search over the space of functions. Therefore, we turn to machine learning algorithms that are designed for this task.

Specifically, we employ the XGBoost algorithm proposed by Chen and Guestrin (2016). XGBoost is one of the most successful prediction algorithms developed in the last few years, and has been widely adopted in both academia and industry.[9] Broadly speaking, boosted trees can be

[9]Boosted trees in general, and XGBoost in particular, perform exceptionally well in tasks involving prediction of human behavior. Examples include store sales prediction, customer behavior prediction, product categorization, ad CTR prediction, course dropout rate prediction, etc. Indeed, almost all the KDD cup winners have used XGBoost as their

thought of as performing gradient descent in function space using shallow trees as the underlying weak learners (Breiman, 1998; Friedman, 2001). While boosted trees have been around for over a decade, Chen and Guestrin (2016)'s implementation is superior to earlier implementation because of four reasons. First, from a methodological standpoint, it can be interpreted as performing Newton boosting in the function space (as opposed to gradient descent), and thereby uses information from the Hessian as well. Thus, both the quality of the leaf structure and the leaf weights learned are more accurate in each step. Second, XGBoost uses a trick commonly used in Random Forests – column sub-sampling,which reduces the correlation between subsequent trees. Third, XGBoost employs a sparsity-aware split finding, which makes the algorithm run faster on sparse data. Finally, from an implementation perspective, XGBoost is highly parallelized, which makes it fast and scalable.

We refer interested readers to Appendix §C a more detailed description of XGBoost and now focus on two key components of our implementation.

4.4.1 Loss Function

To train any learning model, we need to specify how to penalize model fit, i.e., the difference in the observed data y_{i,a_i} and model predictions $\hat{m}_{i,a_i}$, given that ad a is shown in impression i. This is done through a loss function, which the machine learning algorithm minimizes. Given the binary nature of our outcome variable, we use logarithmic loss (log-loss) as our loss function. It is the most commonly used loss function in the CTR prediction literature (Yi et al., 2013) and has some attractive properties, e.g., a faster convergence rate compared to other loss functions such as squared loss (Rosasco et al., 2004). The log-loss for the estimated model can be written as:

$$\mathcal{L}^{log\ loss}(\hat{\mathbf{M}}, \mathbf{y}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_{i,a_i}\log\left(\hat{m}_{i,a_i}\right) + (1-y_{i,a_i})\log\left(1-\hat{m}_{i,a_i}\right)\right) \tag{7}$$

4.4.2 Validation

Validation is an important part of training any machine learning model. The boosting algorithm is designed to continuously update the prediction rule (or current estimate of $f(\cdot)$) to capture more and more complex relationships/interactions between the features $X_{i,a}$, in order to predict $y_{i,a}$. Since we do not impose any assumptions on the parametric form of $f(\cdot)$, this will likely lead to over-fitting, i.e., the model will evolve to fit too closely to the training data and perform poorly out-of-sample. As a result, if we use training data for both model-specification (parameter estimation) and model-selection (hyper-parameter estimation), we might end up picking a model with great in-sample performance but poor out-of-sample performance. Validation helps us avoid

learning algorithm (either as a standalone model or in ensembles) since 2015.

this problem by using parts of the data to validate the model. This ensures that the chosen model, $f(\cdot)$, will have a good out-of-sample performance. Please see Appendix §C.2 for a full description of our validation procedure.

5 Results from the Machine Learning Targeting Models

Recall that the goal of our machine learning framework is to estimate the matrix M defined in Equation (2). As such, our $\hat{M}$ contains CTR estimates for: (1) the ads shown in the data, and (2) counterfactual situations, i.e., ads that could have been shown. In §5.1, we focus on the actual data and present results on the predictive performance of our framework on the observed sample. We also document the contribution of behavioral vs. contextual information to our framework in this section. Next, in §5.2, we focus on the counterfactual estimates in $\hat{M}$ and evaluate the gains in CTR from an efficiently targeting policy. Finally, in §5.3 we discuss robustness and scalability.

5.1 Predictive Performance of the Machine Learning Model

5.1.1 Evaluation Metric

To evaluate whether a targeting model improves our ability to predict clicks, we first need to define a measure of predictive accuracy or an evaluation metric. In line with our loss function, we use "Relative Information Gain" or ***RIG***, which is defined as the percentage improvement in log-loss over the baseline that simply predicts average CTR for all impressions. Formally:

$$RIG(\hat{\mathbf{M}}, \mathbf{y}) = \left(1 - \frac{\mathcal{L}^{log\ loss}(\hat{\mathbf{M}}, \mathbf{y})}{\mathcal{L}^{log\ loss}(\bar{\mathbf{y}}, \mathbf{y})}\right) \times 100, \tag{8}$$

where $\bar{\mathbf{y}}$ is the average CTR in the sample. Average CTR is the simplest aggregate metric available from any data, and using it as the baseline prediction tells us how well we can do without any model. It is important to control for this baseline because if the average CTR is very high (close to 1) or very low (close to zero, as in most e-commerce settings, including ours), a naive prediction based on the average CTR leads to a pretty good log-loss. Normalizing the log-loss with the average CTR reduces the sensitivity of the metric to the data distribution (He et al., 2014). Nevertheless, we need to be careful when interpreting ***RIG***s computed on different datasets because there is no obvious normalization in those cases (Yi et al., 2013).

In Appendix §D.1, we present four other commonly used evaluation metrics – (1) Mean Squared Error, (2) AUC, (3) 0/1 Loss, and (4) Confusion Matrix. We discuss the pros/cons of these metrics and demonstrate the performance of our model on them.

Evaluation Metric	Training and Validation	Test
LogLoss for Full Model	0.041927	0.044364
LogLoss for Baseline Model	0.051425	0.054070
RIG of Full Model	18.47%	17.95%

Table 4: LogLoss and *RIG* (in percentage) shown for training, validation, and test data.

5.1.2 Predictive Accuracy of the Full Targeting Model

We now discuss our framework's ability to predict the actual outcomes in the data. Table 4 shows the gains in prediction for: (1) training and validation data, and (2) test data. The first row depicts the log-loss for the Full model (which uses the set of all features and trains the XGBoost model). The second row depicts the log-loss for the baseline model, which is simply the average CTR for the dataset. The third row is the *RIG* of the Full model compared to the baseline.

The *RIG* of the Full model over the baseline is 17.95% on the test data, a substantial improvement in CTR prediction problems (He et al., 2014). This suggests that the data collected by the ad-network is quite valuable and that our machine learning framework has significant predictive power on whether an impression-ad combination will receive a click.

The *RIG* improvement for training and validation data is 18.47%, which is somewhat higher than 17.95% for the test data. There are two potential reasons for this. First, all statistical models estimated on finite data have higher in-sample fit than out-of-sample fit. Indeed, this is the main reason we use the test data to evaluate model performance. Second, the difference could simply reflect the differences in the underlying data distributions for the two data-sets. As discussed in §5.1.1, we cannot compare *RIG* across data-sets because it is co-determined by the model and data. Thus, the difference between the *RIG* values across the data-sets is not necessarily informative.

5.1.3 Value of Information: Behavioral vs. Contextual Features

We now examine the impact of different types of features on the predictive accuracy of our model. This is important for two reasons. First, data storage and processing costs vary across feature types. For example, some user-specific behavioral features require real-time updating, whereas pure-contextual features tend to be more stable, and can be updated less frequently. In order to decide whether to store and update a feature or not, we need to know its incremental value in improving targeting. Second, the privacy and policy implications of targeting depend on the features used. For example, models that use behavioral features are less privacy-preserving than those that use purely contextual features. Before adopting models that are weaker on privacy, we need objective measures of whether such models actually perform better.

Recall that our features can be categorized into three broad overlapping sets – 1) Behavioral,

RIG over Baseline	**Full Sample**	**Top Ads and Top Apps**	**Filtered Sample**
Behavioral Model	12.14%	14.82%	14.74%
Contextual Model	5.25%	5.98%	6.77%
Full Model	17.95%	22.85%	22.45%
No. of Impressions	9,625,835	6,108,511	4,454,634
% of Test Data	100%	63.5%	46.28%

Table 5: Comparison of Behavioral and Contextual models for different samples of test data.

denoted by F_B, 2) Contextual, denoted by F_C, and 3) Ad-specific, denoted by F_A. We now use this categorization to define two models:

- **Behavioral model:** This model is trained using behavioral and ad-specific features, without including any contextual features. Formally, the feature set used is $(F_B \cup F_A) \setminus F_C$.
- **Contextual model:** This model is trained using only contextual and ad-specific features, without including any behavioral features. The feature set for this model is $(F_C \cup F_A) \setminus F_B$.

Both models include ad-specific features that are neither behavioral nor contextual, e.g., the total impressions received by the ad shown in the impression in the past month (Feature 2 in Table A1 in the Appendix).[10] They also use the same loss function and training algorithm, and only differ on the set of features used. Hence, it is possible for us to directly compare the *RIG* of one model over another within the same data.[11]

The results from these two models and their comparisons with the Baseline model are presented in Table 5. First, consider the results for the full test data (presented in the second column). The Behavioral model has a 12.27% *RIG* over the baseline, which is considerably higher than 5.12%, the *RIG* of the Contextual model over the baseline.[12] Together, these findings suggest that behavioral targeting is more effective compared to contextual targeting in mobile in-app advertising. While it is possible that additional contextual information (that we do not have) can change the relative ordering of these findings, our findings establish a base case for these comparative results.

One possible critique of the above analysis is that it does not exploit the full capacity of contextual information since we treat all the non-top ads as advertiser category and all the non-top apps as one app category during feature generation (see §4.3.1). To address this issue, we consider a

[10]We can also specify Behavioral and Contextual models that ignore ad-specific information. The qualitative results on the relative value of behavioral and contextual information for that case are similar to those presented here.

[11]As discussed in §5.1.1, *RIG*s are not directly comparable across different data-sets. Simply put, in Table 5, comparisons within a column are interpretable, but comparisons across a row are not.

[12]The Behavioral model is also closer in performance to the Full model than the Contextual model. Compared to the Full model, the loss in performance of the Behavioral model is -6.95% and that of the Contextual model is -15.67%.

sub-sample of the test data which only consists of impressions that were shown in a top app and showed a top ad and re-run all the above comparisons. This accounts for 63.5% of our test data. The performance of our Full model on this subset of the data is even better than that on the full sample because there is no information loss on the ads or apps. The findings on the relative value of behavioral vs. contextual features are even stronger in this data-set, which suggests that our results in the full sample were not driven by the lack of good contextual information.

Finally, in the last column, we show the performance of our model on the Filtered Sample (described in §4.2.2), which is the sample that we use for conducting our counterfactual analysis. Our qualitative findings remain the same for this sample too.

5.2 Counterfactual Analysis: Efficiency Gains from CTR-maximizing Targeting Policy

We now focus on an important counterfactual question from the platform's perspective – If the platform employs an efficient targeting policy, such that each impression is allocated the ad with the highest predicted CTR in that impression, to what extent can it improve the CTR in the system?

Recall that τ_0 and τ^* denote the current and efficient targeting policy, as defined in Equations (3) and (5), respectively. We can then use the following equation to calculate the gains in average CTR:

$$\rho(\tau^*, \tau_0; N_F) = \frac{\hat{m}^{\tau^*}}{\hat{m}^{\tau_0}} = \frac{\frac{1}{N_F}\sum_{i=1}^{N_F} \hat{m}_{i,\tau^*(i)}}{\frac{1}{N_F}\sum_{i=1}^{N_F} \hat{m}_{i,\tau_0(i)}}, \tag{9}$$

where N_F is the number of impressions in the Filtered sample. It is crucial to conduct this couterfactual on the Filtered Sample (instead of the Full sample) for the reasons discussed in §4.2.2.

We find that an efficient targeting policy based on our machine learning model increases average CTR by 66.80% over the current regime. This is a substantial improvement, and suggests that targeting based on behavioral and contextual features can lead to significant efficiency gains.

Next, we examine how efficiency-gain varies by impression. Specifically, for each impression we calculate the percentage improvement in CTR with efficient targeting as $\left(\frac{\hat{m}_{i,\tau^*(i)}}{\hat{m}_{i,\tau_0(i)}} - 1\right) \times 100$ and examine the distribution of this metric over impressions. In Figure 7, we show the histogram of this percentage improvement in CTR for the impressions in the Filtered Sample. We find that there is considerable heterogeneity in CTR improvements across impressions. The peak at the left side of the graph (at one) denotes cases where the $\tau_0(i) = \tau^*(i)$, where the platform happened to randomly selected the right ad that maximizes eCTR. Interestingly, we find that the median improvement in CTR is about 105.35%, implying that efficient targeting policy can make over half the impressions twice as clickable as the current system. In sum, we find that an efficient targeting policy leads to significant gains in clicks for the platform.

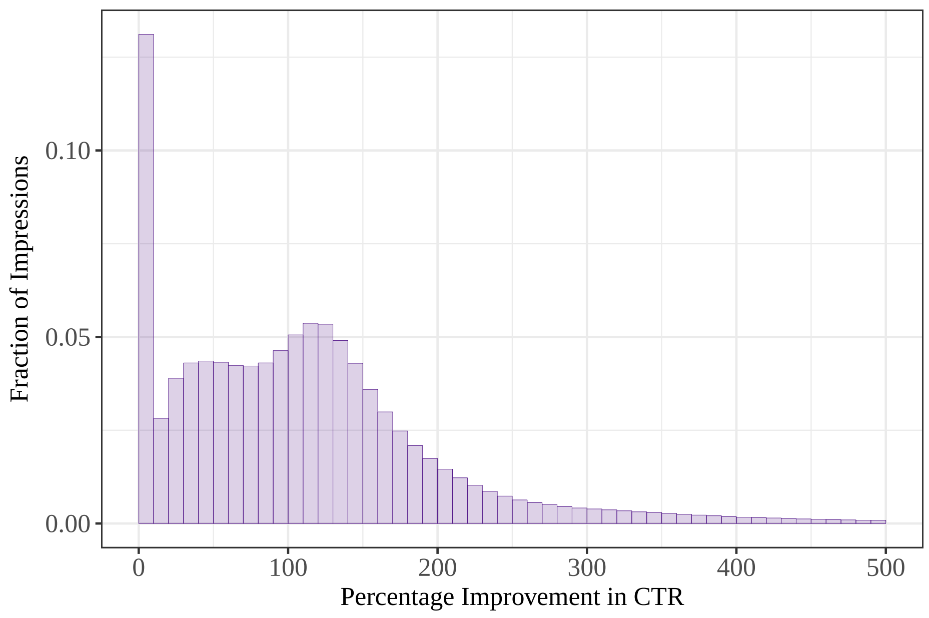


Figure 7: Histogram of percentage improvement in CTR over the current system using the efficient targeting policy

Nevertheless, a key question that remains unanswered is whether an efficient targeting policy is also revenue maximizing for the platform. Therefore, in §6, we incorporate competition and examine the relationship between efficiency and revenues.

5.3 Scalability and Robustness

We perform extensive checks on robustness of all aspects of our machine learning approach and its scalability. We discuss these tests briefly here, and refer readers to Appendix §D for details.

First, in Appendix §D.1, we show that our results are robust even if we use other evaluation metrics (AUC, MSE, 0/1 Loss, and confusion matrix). Second, in Appendix §D.2, we confirm that XGBoost is the best learning algorithm for our prediction task by comparing its performance to five other commonly used algorithms (Least Squares, LASSO, Logistic Regression, Classification And Regression Tree, and Random Forests). Third, in Appendix §D.3, we run a few robustness checks on the feature generation framework by considering alternative ways of aggregating over history as well as app-specific dummies. Again, we find no improvement in the model's predictive performance under these different specifications. Fourth, in Appendix §D.4, we present some checks to establish that our data sample is sufficient and large enough to produce reliable results. Specifically, we find that the *RIG* gains start stabilizing with the sample of 100,000 users, and that our sample of 728,340 users is more than sufficient for our purposes. Finally, in Appendix §D.5, we show that our results are not sensitive to the validation procedure used to pick the tuning parameters by comparing with other methods, e.g., hold-out validation and k-fold cross validation.

6 Analysis of Revenue-Efficiency Trade-off

In §5, we showed that the ad-network can substantially increase CTR with efficient targeting. However, that analysis was silent on the ad-network's incentives to target and agnostic to revenues. In this section, we seek to answer two sets of important questions by focusing on competition and incentives. First, to what extent is the ad-network incentivized to allow targeting and is there an optimal level of targeting from its perspective? Second, how does the total surplus accrued by advertisers vary with targeting levels, and is there heterogeneity in advertisers' preferences on the optimal level of targeting?

Incentives are particularly important in this context because if the platform is incentivized to not allow behaviorally targeted bids, then we may naturally converge to a regime with higher consumer privacy protection. In contrast, if the platform is incentivized to allow behavioral targeting, then an external agency (e.g., government) may have to impose privacy regulations that balance consumers' need for privacy with the platform's profitability motives. Similarly, if a substantial portion of advertisers prefer a more restrictive targeting regime, then the mobile ad-industry can self-regulate. So we seek to quantify the platform's and advertisers' profits under different levels of targeting.

The rest of this section proceeds as follows. In §6.1, we present a simple example to fix ideas and highlight the platform's efficiency-revenue trade-off. In §6.2, we present a stylized analytical model that characterizes the total surplus and platform revenues under different targeting strategies. In §6.3, we take this analytical model to data and present an empirical analysis of auctions with targeting. Finally, in §6.4, we share our findings and conclude with a discussion on optimal mechanism design and the limitations of our findings.

6.1 Revenue-Efficiency Trade-off: A Simple Example

In an important paper, Levin and Milgrom (2010) argue that micro-level targeting can thin auction markets, which in turn can soften competition and make the platform worse off. In Figure 8, we present a simple example to illustrate this idea. In this example, we consider a platform with two impressions and two advertisers whose valuations for these impressions do not align: advertiser 1 has much higher valuation for impression 1 compared to impression 2, whereas the opposite is true for advertiser 2. Assume that the platform uses second price auctions with Cost per Impression (CPI) pricing, where the highest bidder wins the impression and pays the bid of the second-highest bidder. We consider two regimes. In the Full Targeting regime, the platform allows advertisers to submit targeted bids for each impression. In the No Targeting case, advertisers cannot distinguish

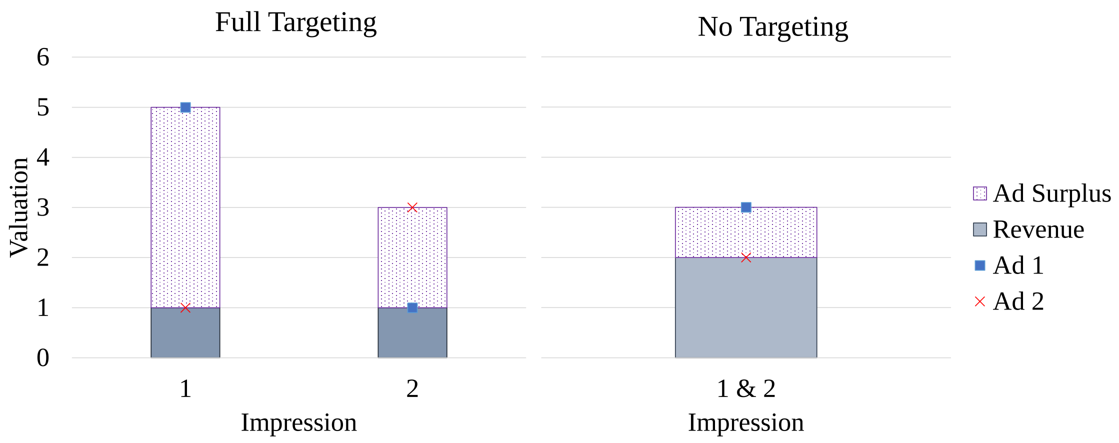


Figure 8: Market outcomes under full vs. no targeting. The platform sells two impressions. Ad 1 and Ad 2 have valuations 5 and 1 for impression 1, and valuation 1 and 3 for impression 2, respectively. When bundled together, advertisers cannot distinguish between ads, giving an aggregate value of 3 and 2 to Ad 1 and 2, respectively. The entire shaded area in each case shows the total surplus generated. The area on the top is the share of advertisers and that in the bottom goes to the platform. See Appendix E for a detailed analysis of this example.

between the two impressions, and therefore have to submit the same bid for both the impressions (i.e., no targeted bidding). As shown in Figure 8, the platform cannot extract sufficient revenue if advertisers can distinguish between impressions (full targeting). However, the platform is able to extract more revenue by not revealing the identity of these impressions since advertisers are forced to rely on their aggregate valuation for both impressions together in this case. This example thus illustrates the platform's trade-off between value creation and value appropriation, and highlights the platform's incentives to limit advertisers' ability to target.

6.2 Analytical Model of Auction with Targeting

We now develop a simple analytical model that captures the trade-offs discussed above. To reflect the idea of narrow targeting and thin markets as envisioned by Levin and Milgrom (2010), we make two modeling choices. First, the idea of revenue loss in thin markets is due to the use of efficient auctions that guarantee that the highest valuation bidder will win. While efficiency is satisfied in many auction mechanisms, we focus on second-price auctions since it is the most commonly used auction in online advertising. Moreover, it has the truth-telling property that makes our analysis more tractable. Second, the idea of narrow targeting by advertisers requires the pricing mechanism to be per-impression. In a cost-per-click mechanism, advertisers do not care about the match value of impressions since they are charged per click. For these reasons, we consider a setting where the platform uses a second-price auction mechanism with CPI pricing. Nevertheless, it is worth noting that neither of these two assumptions are essential to our analysis. Later in this section, we discuss how our results can be extended to other efficient auction mechanisms and/or cost-per-click pricing.

As before, we consider a platform that receives N impressions and serves A advertisers. Let v_{ia} denote ad a's private valuation from impression i, and let V denote the value matrix:

$$V = \begin{bmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,A} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,A} \\ \vdots & \vdots & \ddots & \vdots \\ v_{N,1} & v_{N,2} & \cdots & v_{N,A} \end{bmatrix} \tag{10}$$

If an advertiser a can distinguish between all the impressions, s/he will submit targeted bids for each impression i. In a second-price auction, that is equivalent to a's valuation for impression i, $v_{i,a}$.

However, the extent to which advertisers can target depends on the level of targeting allowed by the platform. If certain information is not disclosed, advertisers may not be able to distinguish two impressions i and j. In such cases, a risk-neutral bidder's valuation for both impressions is the same and is equal to the expected value from the bundle of i and j (Sayedi, 2018). For example, if the platform does not allow targeting at the app level, then advertisers cannot distinguish between impressions in two different apps, and their optimal strategy would be to submit the same bids for the impressions in both apps. Formally:

Definition 3. Let I_l denote the set of impressions in bundle l. A targeting level $\mathcal{I} = \{I_1, I_2, \ldots, I_L\}$ denotes the platform's decision to bundle N impressions into L bundles such that advertisers can only bid for bundles and not impressions within the bundle. As such, impressions are only distinguishable across bundles, but not within a single bundle. That is, for bundle I_j, the advertiser a has the valuation $\frac{1}{|I_j|}\sum_{k \in I_j} v_{ka}$.

This definition characterizes all targeting levels from impression-level targeting to no targeting. Impression-level targeting occurs when each impression is a bundle ($L = N$), i.e., an advertiser can distinguish between all impressions and place targeted bids for each impression. In contrast, no targeting denotes the case where the platform bundles all impressions into one group ($L = 1$) implying that an advertiser can only have one valuation aggregated over all impressions ($\frac{1}{N}\sum_{i=1}^{N} v_{ia}$ for any a). Any intermediate strategy where $1 < L < N$ can be interpreted as partial targeting. An example of partial targeting is app-level targeting, where each bundle is an app and impressions are distinguishable across apps but not within apps.

We can characterize the relative granularity of two targeting levels as follows:

Definition 4. Let $\mathcal{I}^{(1)}$ and $\mathcal{I}^{(2)}$ denote two targeting levels such that $\mathcal{I}^{(1)} = \{I_1^{(1)}, \ldots, I_{L_1}^{(1)}\}$ and $\mathcal{I}^{(2)} = \{I_1^{(2)}, \ldots, I_{L_2}^{(2)}\}$. Targeting level $\mathcal{I}^{(1)}$ is at least as granular as $\mathcal{I}^{(2)}$ if for any $I_j^{(1)} \in \mathcal{I}^{(1)}$, there exists a $I_k^{(2)} \in \mathcal{I}^{(2)}$ such that $I_j^{(1)} \subseteq I_k^{(2)}$. In words, if two impressions i and j are distinguishable in $\mathcal{I}^{(2)}$, then they will be distinguishable in $\mathcal{I}^{(1)}$.

We can use this definition to compare the granularity of two targeting levels. For example, app-user level targeting is more granular than app level targeting. Now, the main question that the platform faces is at what level of granularity they should disclose information and allow targeting. Since we focus on the second-price auction, the highest-bidding ad in any impression wins that impression, and pays second-highest bid. This auction also guarantees truth-telling property, i.e., for each bundle, advertisers submit their aggregate valuation for that bundle as derived in Definition 3. The following proposition determines the relationship between the granularity level of targeting and market outcomes such as surplus and revenue:

Proposition 1. Consider two targeting levels $\mathcal{I}^{(1)}$ and $\mathcal{I}^{(2)}$ such that $\mathcal{I}^{(1)}$ is at least as granular as $\mathcal{I}^{(2)}$. Let $S^{(j)}$ and $R^{(j)}$ denote the total surplus and platform's revenue under targeting level $j \in \{1, 2\}$. Then, for any distribution of valuations: $S^{(1)} \geq S^{(2)}$, but there is no fixed relationship between $R^{(1)}$ and $R^{(2)}$.

Proof. See Appendix F. □

As the granularity of targeting increases, the total surplus generated increases, but the platform's revenue can go in either direction (unless we impose strong distributional assumptions on match values). Thus, while the matches are more efficient with more granular targeting, the platform may not be able to appropriate these efficiency gains. It is worth emphasizing the our analysis of revenue and surplus holds for any efficient auction, given Revenue Equivalence Theorem (Myerson, 1981; Riley and Samuelson, 1981). Further, in Appendix §G, we show that the same qualitative findings hold for a cost-per-click pricing mechanism.

6.3 Empirical Analysis of Auctions with Targeting

We now take this analytical model to data and examine market outcomes under different targeting levels. The first step is to obtain estimates for all the elements in matrix V defined in Equation (10). We can write:

$$v_{i,a} = v_a^{(c)} m_{i,a}, \tag{11}$$

where $v_a^{(c)}$ is private valuation ad a gets from a click and $m_{i,a}$ is the match valuations or expected CTR of ad a if shown in impression i. While we cannot directly estimate $v_{i,a}$ from the data, we can estimate both elements on the right-hand side – click and match valuations for advertisers. We first discuss how we approximate advertisers' click valuations in §6.3.1. Next, in §6.3.2, we explain how we can use our estimated match value matrix $\hat{M}$ from §4 to derive advertisers' match values for different targeting regimes. Finally, in §6.3.3, we discuss our startegy to estimate the expected surplus, platform revenues, and advertisers' surplus.

6.3.1 Estimation of Click Valuations from Observed Bids

Here, our goal is to link advertisers' observed bids to their private click valuations. In quasi-proportional auctions, advertisers do not bid their valuations in equilibrium (unlike second-price auctions). Therefore, we need to first solve for the advertisers' bidding strategy given our current auction mechanism and then invert it to get valuations. (Mirrokni et al., 2010) establishes the existence and uniqueness of a pure strategy Nash equilibrium for a class of quasi-proportional auctions. Using their results, we can derive the following approximation for our case:

$$\hat{v}_a^{(c)} \approx 2b_a^*, \tag{12}$$

where b_a^* is ad a's observed bid. This is a direct result of Equation (5.2) in Mirrokni et al. (2010), when there are many bidders and no one has a disproportionately high chance of winning. Later in §H.1, we propose alternative methods to estimate valuations in a quasi-proportional auction and show that our results are robust to these alternatives.

6.3.2 Recovering Match Values

We now discuss how we can use our estimate of matrix M from our targeting framework to recover match values for any targeting regime.

$m_{i,a}$ is ad a's match value for any impression i, if all impressions are distinguishable to her and she is competing for that impression. This follows naturally from our arguments on the accuracy of match value estimates in §4.2.1. However, if two impressions are not distinguishable, the advertiser needs to use the aggregate estimate for that bundle. That is, for any targeting level $\mathcal{I} = \{I_1, I_2, \ldots, I_L\}$, we can write the match value of advertiser a for impression i in bundle $\mathcal{I}$, $m_{i,a}^{\mathcal{I}}$, as follows:

$$\hat{m}_{i,a}^{\mathcal{I}} = \sum_{j=1}^{L} \mathbb{1}(i \in I_j) \frac{\sum_{k \in I_j} \hat{m}_{k,a} e_{k,a}}{\sum_{k \in I_j} e_{k,a}} \quad \forall \;\; i \in \mathcal{I}, \tag{13}$$

where $e_{k,a}$ are elements of the filtering matrix that allows us to disregard inaccurate estimates and take the average of the rest.

Here, we assume that for any targeting-level $\mathcal{I}$, advertisers can infer their private match values for the bundles at that targeting level $\hat{m}_a^{\mathcal{I}}$. This is reasonable because if the platform allows impression-level targeting, the platform would automatically share the impression-level data of each advertiser a with that advertiser (but not other advertisers). If a has sufficient data, then a can accurately estimate the match value vector $m_{i,a}$ for impression i from their own data. Similarly, if the platform only allows targeting at level $\mathcal{I}$, then advertisers would automatically have information on which bundle an impression belongs to as well as outcomes (whether impressions in a given bundle received clicks or not), and can therefore accurately infer their match values at the granularity of the bundle. While this assumption always holds from a theoretical standpoint, it may not hold in practice because advertisers need sufficient data to obtain accurate estimates of their match values. Thus, the match value estimates of smaller advertisers and/or new advertisers can be noisy (though they will be consistent). Later, in §H.2, we show that our findings are robust even in situations where advertisers' match value estimates are noisy/imperfect.

Finally, the match value estimates derived from our quasi-proportional auction are assumed to remain the same under a second-price auction. This is reasonable because the match value simply indicates the click probability of a user in a given context for an ad. There is no economic rationale for users' click behavior to be a function of the auction, especially since users often do not know which auction is running at the back-end. Intuitively, click valuations and match valuations are treated as structural parameters.

6.3.3 Estimation of Revenue and Surplus

Given our estimates for click and match valuations, we can obtain estimates of the elements of the valuation matrix V as $\hat{v}_{i,a} = \hat{v}_a^{(c)} \hat{m}_{i,a}$. Further, we can estimate advertisers' expected value of impression i under targeting granularity (or level) $\mathcal{I}$ as $\hat{v}_{i,a}^{\mathcal{I}} = \hat{v}_a^{(c)} \hat{m}_{i,a}^{\mathcal{I}}$. We now discuss our procedure to estimate revenue and surplus for any targeting level $\mathcal{I}$. First, we determine the winners of each impression as follows:

$$\hat{a}_i^*(\mathcal{I}) = \underset{a}{\operatorname{argmax}}\, \hat{v}_{i,a}^{\mathcal{I}} e_{i,a}, \tag{14}$$

where $\hat{a}_i^*(\mathcal{I})$ is the winner for impression i under targeting regime $\mathcal{I}$. Note that the multiplication by the element of the filtering matrix $e_{i,a}$ simply ensures that the ad is competing in the auction for impression i and that the counterfactual match value estimates are valid, as discussed in §4.2.2.

While the winner is determined using the advertisers' expected value of impression i under a specific targeting regime, surplus is calculated using the actual valuation matrix because it denotes the expected value that would be realized in the system if advertiser $\hat{a}_i^*(\mathcal{I})$ is allocated impression i.

So we can write the surplus under targeting granularity $\mathcal{I}$ as:

$$\hat{S}^{\mathcal{I}} = \sum_{i=1}^{N^F} \hat{v}_{i,\hat{a}_i^*(\mathcal{I})} \tag{15}$$

To estimate the platform revenues, however, we need to use advertisers' expected values under targeting level $\mathcal{I}$, as these values guide their bidding behavior. Further, we need to incorporate the fact that the revenue generated from impression i is the second highest bid (or valuation) for it. Thus, the revenue under $\mathcal{I}$ is:

$$\hat{R}^{\mathcal{I}} = \sum_{i=1}^{N^F} \max_{a \backslash \hat{a}_i^*(\mathcal{I})} \hat{v}_{i,a}^{\mathcal{I}} e_{i,a} \tag{16}$$

Finally, we can estimate advertiser a's surplus under targeting level $\mathcal{I}$ as follows:

$$\hat{W}_a^{\mathcal{I}} = \sum_{i=1}^{N^F} \left(\hat{v}_{i,\hat{a}_i^*(\mathcal{I})} - \max_{a \backslash \hat{a}_i^*(\mathcal{I})} \hat{v}_{i,a}^{\mathcal{I}} e_{i,a} \right) \mathbb{1}\left(\hat{a}_i^*(\mathcal{I}) = a \right) \tag{17}$$

Note that the estimation is carried out on the Filtered Sample to ensure that our match value estimates are accurate, and hence the averaging in the above equations is done over N_F.

6.4 Counterfactual Results and Privacy Implications

While we can analyze market outcomes for any targeting level, we focus on the following four targetimg regimes that have a one-to-one correspondence with our analysis in §5.1.3:

- No targeting ($\mathcal{I}^{(\mathcal{N})}$) – The platform allows no targeting. As such, there is only one bundle (which constitutes all impressions) and advertisers cannot distinguish between any impressions.
- Contextual targeting ($\mathcal{I}^{(\mathcal{C})}$) – The platform only allows contextual targeting. Here, advertisers can distinguish between impressions in different contexts (app and time). However, impressions from different users in the same context is not distinguishable.
- Behavioral targeting ($\mathcal{I}^{(\mathcal{B})}$) – The platform allows behavioral targeting, thereby allowing advertisers to distinguish between users but not contexts. Here, advertisers can submit bids targeted at the user-level, but cannot distinguish two impressions by the same user in different contexts.
- Full targeting ($\mathcal{I}^{(\mathcal{F})}$) – Platform allows impression-level targeting, i.e., each impression is a bundle and therefore distinguishable. Advertisers can submit targeted bids for each impression.

Using Proposition 1, we can show that $S^{(\mathcal{F})} \geq S^{(\mathcal{N})}$, and that $S^{(\mathcal{C})}$ and $S^{(\mathcal{B})}$ lie in between since both Contextual and Behavioral targeting can be interpreted as imperfect targeting. However, we cannot theoretically pin down their relative magnitudes because these two types of information are

Targeting	**Total Surplus**	**Platform Revenue**	**Advertisers' Surplus**
Full	9.45	8.35	1.10
Behavioral	9.18	8.35	0.84
Contextual	8.99	8.44	0.55
No targeting	8.36	8.30	0.06

Table 6: Platform revenues, advertisers' surplus, and total surplus for different levels of targeting. The numbers are reported in terms of the average monetary unit per impression.

orthogonal (one cannot be interpreted as more granular than the other). So we can only show that: $S^{(\mathcal{F})} \geq S^{(\mathcal{C})}, S^{(\mathcal{B})} \geq S^{(\mathcal{N})}$. Further, we have no theoretical guidance on which of these targeting regimes maximizes platform revenues. We therefore use the empirical framework described in §6.3 to derive estimates of platform revenue, advertisers' surplus, and total surplus under the four targeting regimes, and empirically answer the question: "What is the optimal level of targeting that maximizes the platform's revenue?". The results from this exercise are shown in Table 6.

6.4.1 Platform's Revenue

Consistent with our theory model, our empirical results suggest that more granular targeting leads to higher efficiency in the market: the total surplus under full targeting is 13.02% higher than the no targeting case. Further, in line with our findings in §5.1.3, we find that the total surplus under behavioral targeting is 2.13% higher than the contextual targeting. However, platform revenues exhibit more of an inverted U-shaped curve. They are maximized when the platform restricts targeting to the contextual level. When the platform allows behavioral targeting, advertisers achieve a greater ability to target. While this increases the total surplus in the system, much of this surplus is appropriated by advertisers and the platform's revenue suffers.[13] Thus, the platform's incentives are not perfectly aligned with that of advertisers. Indeed, the platform's optimal targeting level is privacy preserving and aligned with consumers' preferences. We thus find support for the advertising industry's claim that external regulation is not necessary to reduce user-tracking/targeting, and that the industry can self-regulate.

Our findings give rise many interesting suggestions/ideas on optimal mechanism design and information revelation from the platform's perspective. Limiting targeting to the contextual-level is an obvious strategy. However, this approach also reduces the total surplus and hence caps the platform's revenues. Thus, the optimal path for the platform may not be to restrict targeting, but instead to consider mechanisms that can do both – increase efficiency and extract the revenue from

[13]Nevertheless, our findings are weaker than those predicted by theory models, i.e., while revenues reduce with more granular targeting, the drop is not very large. This suggests that the strong distributional assumptions on the match values in earlier theory papers (e.g., Hummel and McAfee (2016)) may not hold in real ad auctions.

To \ From	**Full**	**Behavioral**	**Contextual**	**Baseline**
Full	NA	23	33	35
Behavioral	14	NA	34	36
Contextual	4	3	NA	33
Baseline	2	1	4	NA

Table 7: Number of advertisers who benefit by moving from one targeting regime (column) to another (row) (out of 37 top advertisers).

winning advertisers by shrinking the informational rent. For instance, the platform could allow behavioral targeting, and also adopt the standard theoretical solution proposed for revenue extraction – optimal reserve prices (Myerson, 1981; Riley and Samuelson, 1981). Ostrovsky and Schwarz (2016) validate these theoretical findings using field experiments for search ads. However, they only consider optimal reserve prices for broad sets of keywords and assume eCTRs to be homogeneous across advertisers. In contrast, we have a setting where each impression is a unique product and advertisers' match values for an impression are heterogeneous. So, in our case, the platform has to develop a system that can set dynamic impression-specific optimal reserve prices.

6.4.2 Advertisers' Surplus

We begin by comparing total advertiser surplus across the four targeting regimes. As shown in Table 6, advertiser's surplus is increasing with more granular targeting. This validates our theoretical prediction that more granular targeting helps advertisers by allowing them to generate more accurate estimates of their match values and place targeted bids. Under full targeting, advertisers' surplus is 11.69% of the total surplus whereas this share drops to 0.69% when no targeting is allowed. Further, the share of advertisers' surplus under behavioral targeting is 8.99% which is considerably higher than 6.13%, their share under contextual targeting. Together these findings emphasize the value of behavioral information for advertisers.

Next, we explore whether all advertisers benefit when their ability to target is enhanced. In a competitive environment, greater ability to target does not necessarily translate into higher profits. Instead, it is the ability to target relative to competitors that matters. In Table 7, we show how many advertisers benefit as we move from one targeting regime (column) to another (row).

In general, more advertisers benefit when the platform allows more granular targeting, especially when it allows behavioral targeting. Moving from behavioral, contextual, and no targeting to full targeting benefits 23, 33, and 35 advertisers respectively (first row of Table 7). However, more granular targeting is not uniformly better for all advertisers. The first column of Table 7 depicts situations where advertisers go from the most granular to less granular targeting levels. Interestingly, it is populated with positive numbers, which suggests that some advertisers actually benefit from less granular targeting. For example, there are 14 advertisers who prefer behavioral targeting to full targeting. Similarly, while the majority of advertisers prefer behavioral targeting, there is a small portion of advertisers (3) who prefer contextual targeting. We present a simple example to highlight the intuition behind this – a nutrition supplement ad that advertises on a fitness app can get all the slots in that app at a low cost because other advertisers would place low bids when only app-level targeting is allowed. However, this ad would be worse off if only behavioral targeting is allowed, because the competition for users in this app becomes more intense and this ad will no longer be able to extract a large informational rent.

In sum, our findings offer some evidence that advertisers are likely to be differentially affected by privacy regulation on user-tracking and behavioral targeting. Further research on the sources of heterogeneity in advertisers' incentives can help regulators craft the appropriate privacy policies.

6.5 Robustness Checks

We run a series of robustness checks on two the main components of our estimation – click valuations and match valuations. We describe these tests briefly here. Please see Appendix §H for details.

First, in Appendix §H.1, we consider alternative approaches to estimate click valuations from observed bids and show the robustness of our results. Second, in Appendix §H.2.1, we show the robustness of our results when we add identically distributed noise to all the match value estimates (to reflect the cases where advertisers realize a noisy version of match value estimates from our ML framework). Finally, in §H.2.2, we consider the cases in which the distribution of noise added to match values are different such that larger advertisers have less noisy estimates. We show that our results are robust to these situations.

6.6 Limitations

While we have tried to make our analysis as exhaustive and complete as possible, our counterfactual results should nevertheless be interpreted cautiously with the necessary caveats. First, we assume that advertisers' enhanced ability to target is only reflected in their targeted bidding. In reality however, there might be value creation through other decision variables as well. For example, advertisers can change the design and creatives of their ads and increase their match values, since they can customize their design for a narrower target. This is beyond the scope of our counterfactual analysis since we do not have data on ad identities and banners.[14] Second, we assume that the set of

[14]In our setting, we find believe this is unlikely to contribute much to the value creation because of two reasons. First, in the current regime, we do not observe such actions by advertisers, despite the fact that they are already allowed to run different campaigns on the current targeting areas. This suggests that costs of such actions probably outweigh their benefits. Second, it is worth noting that this is an in-app advertising setting and most ads are apps themselves. As such, ads are mostly informative in nature, and manipulations such as design change may not be very effective.

ads competing for an impression will not change under different targeting regimes. This implies that there is no entry of new ads or exit of existing ads for an impression. While this assumption may not be realistic, it is unlikely to change the qualitative findings of this paper.[15] Third, when estimating revenues and surpluses under different targeting regimes, we use the same distribution of features as seen in the current data. However, the joint distribution of features observed in the data may change under a different targeting strategy because the impressions shown to users and their click behavior is a function of targeting. Thus, the platform's surplus and the machine learning targeting models can both co-evolve in the long run equilibrium. As such, we do not take a strong stance on the optimal level of targeting in long-run equilibrium. Rather, our goal is to provide some empirical evidence in support of the idea that the platform has some incentives to limit behaviorally targeted ads. Moreover, while we run extensive robustness check to ensure the validity of our counterfactual results, these findings nevertheless should be treated as a limited or short-run counterfactuals, which offer some proof of existence for the possibility of self-regulation in this market in light of revenue-efficiency trade-off. In the long-run, many other environmental factors could change, which are outside of the scope of our analysis.

7 Conclusions

Mobile in-app advertising is growing in popularity. In-app ads have unique tracking properties: they allow advertisers and ad-networks to access the device ID of users' mobile devices, and thereby enable high quality behavioral targeting. While this has made them appealing to advertisers, consumers privacy advocates are concerned about their invasiveness. Therefore, marketers and policy-makers are interested in understanding the relative effectiveness of behavioral targeting compared to contextual targeting (which is privacy-preserving, the incentives of ad-networks to engage in behavioral targeting, and the role of regulation in preserving privacy.

We propose a unified framework that consists of two components – a machine learning framework for targeting and an analytical framework for targeting counterfactuals when considering the competition in the market. We apply our framework to data from the leading in-app ad-network of an Asian country. Our machine learning model achieves a *RIG* of 17.95% over the baseline when we evaluate it on test data. This translates to a 66.80% increase in the average CTR over the

[15]There can be two types of changes in the set of ads under a new targeting regime – (1) some ads that are competing for impression i in the current regime will exit the competition, and (2) some ads that are not competing for impression i will enter the competition. The former can only strengthen our findings as it will thin out the market to a greater extent. The latter can be an issue if an ad which is not participating in the auction for i chooses to enter the market. If such an ad ends up in the top two highest valuations it can change our revenue/surplus estimation. To address this issue, we re-derived all the empirical results for the revenue-efficiency analysis shown in Table 6 for the case of full entry (i.e., all 37 ads are competing for all impressions). We found that the substantive results remain the same in this case. Thus, even under the most unfavorable assumption on entry, the qualitative results remain unchanged.

current system if we were to deploy an efficient targeting policy based on our machine learning framework. These gains mainly stem from behavioral information and the value of contextual information is relatively small. Next, we build an analytical model of targeting and theoretically prove that although total surplus grows with more granular targeting between the ad-network and advertisers, the ad-network's revenues are non-monotonic in the granularity of targeting. We then take our analytical model to data and conduct a series of targeting counterfactuals and show that the platform prefers to not allow behavioral targeting. There is also some heterogeneity among advertisers' on their preferred level of targeting. Our findings suggest the ad-network may have natural incentive to naturally preserve users' privacy.

Our paper makes several contributions to the literature. From a methodological standpoint, we propose a unified framework for targeting that exploits the randomization in the allocation of ads to generate accurate counterfactual estimates under various targeting regimes. From a substantive perspective, our paper provides new insights on contextual and behavioral targeting. To our knowledge, this is the first paper to study both revenue and efficiency under these two types of targeting. Finally, from a policy point-of-view, we examine the incentives to target for the two major parties in the advertising eco-system: the platform and advertisers. We expect our model and findings to speak to the debate on privacy regulations in the advertising industry.

References

A. Acquisti, C. Taylor, and L. Wagman. The Economics of Privacy. *Journal of Economic Literature*, 54(2): 442–92, 2016.

W. Amaldoss, K. Jerath, and A. Sayedi. Keyword Management Costs and "Broad Match" in Sponsored Search Advertising. *Marketing Science*, 35(2):259–274, 2015.

S. Athey and D. Nekipelov. A structural model of sponsored search advertising auctions. In *Sixth Ad Auctions Workshop*, volume 15, 2010.

D. Bergemann and A. Bonatti. Targeting in Advertising Markets: Implications for Offline Versus Online Media. *The RAND Journal of Economics*, 42(3):417–443, 2011.

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

L. E. Celis, G. Lewis, M. Mobius, and H. Nazerzadeh. Buy-It-Now or Take-a-Chance: Price Discrimination Through Randomized Auctions. *Management Science*, 60(12):2927–2948, 2014.

O. Chapelle, E. Manavoglu, and R. Rosales. Simple and Scalable Response Prediction for Display Advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):61, 2015.

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.

A. De Corniere and R. De Nijs. Online Advertising and Privacy. *The RAND Journal of Economics*, 47(1): 48–72, 2016.

D. Dzyabura and H. Yoganarasimhan. Machine Learning and Marketing. In *Handbook of Marketing Analytics*. Edward Elgar Publishing, 2018.

J. Edwards. Apple Has Quietly Started Tracking iPhone Users Again, And It's Tricky To Opt Out, 2012. URL `http://www.businessinsider.com/ifa-apples-iphone-tracking-in-ios-6-2012-10`.

A. Edwards-Levy and D. Liebelson. Even Trump Voters Hate This Bill He Just Signed, 2017. URL `http://www.huffingtonpost.com/entry/trump-online-privacy-poll_us_58e295e7e4b0f4a92`

eMarketer. Digital Ad Spending 2019 , 2019. URL `https://content-na1.emarketer.com/us-digital-ad-sp`

J. H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, pages 1189–1232, 2001.

H. Fu, P. Jordan, M. Mahdian, U. Nadav, I. Talgam-Cohen, and S. Vassilvitskii. Ad Auctions with Data. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 184–189. IEEE, 2012.

A. Goldfarb. What is Different About Online Advertising? *Review of Industrial Organization*, 44(2):115–129, 2014.

A. Goldfarb and C. Tucker. Advertising Bans and the Substitutability of Online and Offline Advertising. *Journal of Marketing Research*, 48(2):207–227, 2011a.

A. Goldfarb and C. Tucker. Online Display Advertising: Targeting and Obtrusiveness. *Marketing Science*, 30 (3):389–404, 2011b.

X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, et al. Practical Lessons from Predicting Clicks on Ads at Facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–9. ACM, 2014.

P. Hummel and R. P. McAfee. When Does Improved Targeting Increase Revenue? *ACM Transactions on Economics and Computation (TEAC)*, 5(1):4, 2016.

G. A. Johnson. The Impact of Privacy Policy on the Auction Market for Online Display Advertising. 2013.

J. Kint. Opinion: Europe's Strict New Privacy Rules Are Scary but Right, 2017. URL `http://adage.com/article/digitalnext/europe-s-strict-privacy-rules-terrifying-app`

J. Levin and P. Milgrom. Online Advertising: Heterogeneity and Conflation in Market Design. *The American Economic Review*, 100(2):603–607, 2010.

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

V. Mirrokni, S. Muthukrishnan, and U. Nadav. Quasi-Proportional Mechanisms: Prior-Free Revenue Maximization. In *Latin American Symposium on Theoretical Informatics*, pages 565–576. Springer, 2010.

R. B. Myerson. Optimal Auction Design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

M. Ostrovsky and M. Schwarz. Reserve Prices in Internet Advertising Auctions: A Field Experiment. Working Paper, 2016.

J. G. Riley and W. F. Samuelson. Optimal Auctions. *The American Economic Review*, 71(3):381–392, 1981.

L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–1076, 2004.

M. Sands. How The Cookies Crumble In A Mobile-First World, 2015. URL `https://martechtoday.com/cookies-crumble-mobile-first-world-154114`.

A. Sayedi. Real-Time Bidding in Online Display Advertising. *Marketing Science*, 2018.

O. Toubia, T. Evgeniou, and J. Hauser. Optimization-based and Machine-learning Methods for Conjoint Analysis: Estimation and Question Design. In A. Gustafsso, A. Herrmann, and F. Huber, editors, *Conjoint Measurement: Methods and Applications*, pages 231–258. Springer, 4 edition, 2007.

C. E. Tucker. Social Networks, Personalized Advertising, and Privacy Controls. *Journal of Marketing Research*, 51(5):546–562, 2014.

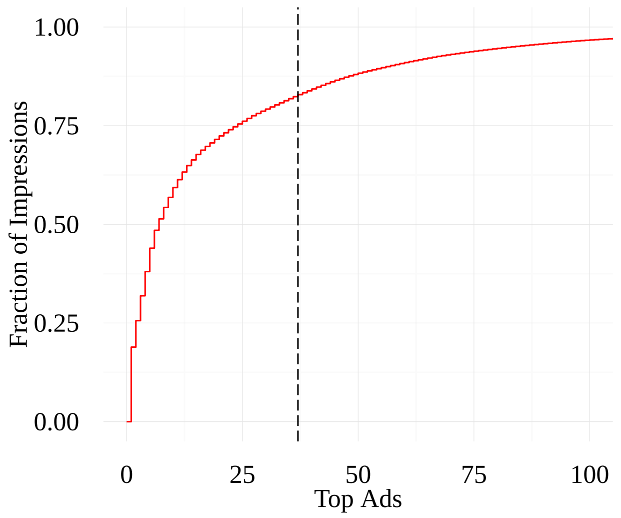
S. Yao and C. F. Mela. A Dynamic Model of Sponsored Search Advertising. *Marketing Science*, 30(3): 447–468, 2011.

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.

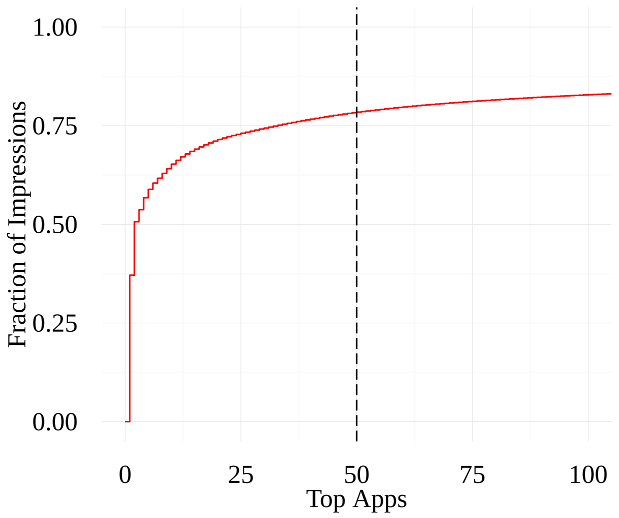
H. Yoganarasimhan. Search Personalization Using Machine Learning. *Forthcoming, Management Science*, 2019.

Appendices

A Summary Statistics



(a) Ads



(b) Apps

Figure 9: Cumulative fraction of impressions associated with the top 100 ads and top 100 apps.

B Feature Generation Framework

We need a set of informative features that can help accurately predict the probability of click for a given impression. Given our research agenda, our features should be able to capture the contextual and behavioral information associated with an impression over different lengths of history preceding the impression (long-term, short-term, and session-level). Therefore, we adopt the main ideas from the functional feature generation framework proposed by Yoganarasimhan (2019). In §B.1, we describe the inputs to the feature functions, in §B.2, we describe the functions, and finally in §B.3, we present the full table of features and their classification.

B.1 Inputs for Feature Functions

Each impression i in our training, validation, and test data can be uniquely characterized by the following four variables:

- the ad a_i which was shown in the impression, where $a_i \in A = \{a^{(1)}, a^{(2)}, \dots, a^{(37)}, a^{(s)}\}$. The elements $a^{(1)}, a^{(2)}, \dots, a^{(37)}$ refer to the identities of the top 37 ads, whereas all the smaller ads are grouped into one category, which is denoted by as $a^{(s)}$. A denotes the set of all ads.

- the app p_i within which it occurred, where $p_i \in P = \{p^{(1)}, p^{(2)}, \dots, p^{(50)}, p^{(s)}\}$. $p^{(1)}$ through $p^{(50)}$ refer to the top 50 apps, and $p^{(s)}$ refers to the category of all smaller apps (which we do not track individually).
- the hour of the day during which the impression occurred (denoted by t_i). t_i can take 24 possible values ranging from 1 through 24 and the set of these values is: $T = \{1, 2, \dots, 24\}$.
- the user u_i who generated the impression, where $u_i \in U = \{u^{(1)}, \dots, u^{(728,340)}\}$. U is thus the full sample of users in the training, validation, and test data.

We also have an indicator for whether the impression was clicked or not (denoted by y_i). In addition, we have a history of impressions and clicks observed in the system prior to this impression.

To generate a set of features for a given impression i, we use feature functions that take some inputs at the impression level and output a corresponding feature for that impression. Our feature functions are typically of the form $F(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie})$ (though some functions may have fewer inputs). We now define each of these inputs:

1. *Ad related information*: The first input θ_{ia} captures the ad related information for impression i. It can take two possible values: $\theta_{ia} \in \Theta_{ia} = \{\{a_i\}, A\}$. If $\theta_{ia} = \{a_i\}$, then the feature is specific to the ad shown in the impression. Instead, if $\theta_{ia} = A$, then it means that the feature is not ad-specific and is aggregated over all possible ads.

2. *Contextual information*: The second and third inputs θ_{ip} and θ_{it} capture the context (where and when) for impression i.

   (a) $\theta_{ip} \in \Theta_{ip} = \{\{p_i\}, P\}$ can take two possible values. If $\theta_{ip} = \{p_i\}$, then the feature is specific to the app where the impression was shown. Instead, if $\theta_{ip} = P$, the feature is aggregated over all apps.

   (b) $\theta_{it} \in \Theta_{it} = \{\{t_i\}, T\}$ characterizes the time-related information for impression i. If $\theta_{it} = \{t_i\}$, then the feature is specific to the hour during which the impression occurred. Instead, if $\theta_{it} = T$, the feature is aggregated over all hours of the day.

3. *Behavioral information*: The fourth input, $\theta_{iu} \in \Theta_{iu} = \{\{u_i\}, U\}$, captures the behavioral information for impression i. If $\theta_{iu} = \{u_i\}$, the feature is specific to user u_i who generated the impression. Instead, if $\theta_{iu} = U$, the feature is aggregated over all users.

4. *History*: The last two inputs, η_{ib} and η_{ie}, capture the history over which the function is aggregated. η_{ib} denotes the **b**eginning or starting point of the history and η_{ie} denotes the **e**nd-point of the history.

(a) $\eta_{ib} \in \mathcal{H}_{ib} = \{l, s, o_i\}$ can take three possible values which we discuss below:

- $\eta_{ib} = l$ denotes **l**ong-term history, i.e., the starting point of the history is September 30, 2015, the date from which data are available.
- $\eta_{ib} = s$ denotes **s**hort-term history, i.e., only data on or after October 25, 2015, are considered.
- $\eta_{ib} = o_i$ denotes **o**ngoing session-level history, i.e., the history starts from the beginning of the session within which the impression occurs.[16] This history is the most accessible in the user's short-term memory. Note that by definition, $\eta_{ib} = o_i$ implies that the function is calculated at the user level.

(b) $\eta_{ie} \in \mathcal{H}_{ie} = \{g, r_i\}$ can take two possible values which we define them as follows:

- $\eta_{ie} = g$ implies that the feature is calculated over the **g**lobal data, i.e., data after October 27, 2015, are not considered. These features are calculated without using any impression from the training, validation and test data-sets.
- $\eta_{ie} = r_i$ refers to **r**eal-time aggregation, i.e., the feature is calculated over all the information up till the focal impression i.

In Figure 10, we present a picture with five example users to illustrate different types of history.

B.2 Feature Functions

We now use the nomenclature described above to define the following functions.

1. *Impressions*$(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie})$: This function counts the number of impressions with the characteristics given as inputs over the specified history.

$$\textit{Impressions}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \sum_{j \in [\eta_{ib}, \eta_{ie}]} \mathbb{1}(a_j \in \theta_{ia}) \mathbb{1}(p_j \in \theta_{ip}) \mathbb{1}(t_j \in \theta_{it}) \mathbb{1}(u_j \in \theta_{iu}), \tag{A.18}$$

where j denotes an impression whose time-stamp lies between the starting point of the history η_{ib} and end point of the history η_{ie}.

For example, *Impressions*$(\{a_i\}, \{p_i\}, \{t_i\}, \{u_i\}; l, r_i)$ returns the number of times ad a_i is shown to user u_i while using app a_i at the hour of day t_i in the time period starting September

[16]A session ends when there is a five-minute interruption in a user's exposure to the ads. So if the time difference between two consecutive impressions shown to a user-app combination is more than five minutes, we assume that the two impressions belong to different sessions.

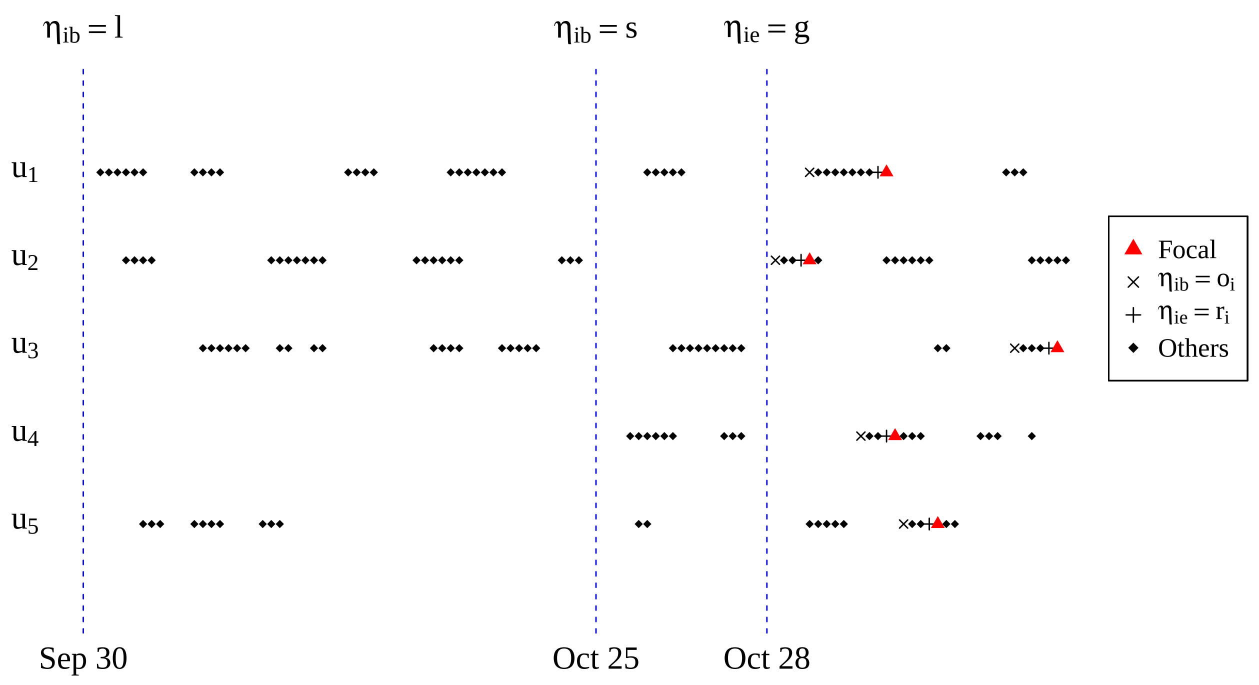


Figure 10: Depiction of history for five example users.The ▲ refers to the focal impression i for which the features are being generated. $+$ denotes the last impression just before the focal impression, and $\times$ refers to the first impression in the session in which the focal impression occurs.

30, 2015, and ending with the last impression before i. Instead, if we are interested in the number of times that user u_i has seen ad a_i over all apps and all hours of the days from October 25, 2015, ($\eta_{ib} = s$) till the end of the global data ($\eta_{ie} = g$), then we would have:

$$\begin{aligned} \textit{Impressions}(\{a_i\}, P, T, \{u_i\}; s, g) &= \sum_{j\in[s,g]} \mathbb{1}(a_j \in \{a_i\})\mathbb{1}(p_j \in P)\mathbb{1}(t_j \in T)\mathbb{1}(u_j \in \{u_i\}) \\ &= \sum_{j\in[s,g]} \mathbb{1}(a_j \in \{a_i\})\mathbb{1}(u_j \in \{u_i\}) \end{aligned}$$

Intuitively, the *Impressions* function captures the effects of repeated ad exposure on user behavior, and ad exposure has been shown to yield higher ad effectiveness in the recent literature (Sahni, 2015; Johnson et al., 2016). In some cases, it may also capture some unobserved ad-specific effects, e.g., an advertiser may not have enough impressions simply because he does not have a large budget.

2. *Clicks*$(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie})$: This function counts the number of clicks with the characteristics given as inputs over the history specified. It is similar to the *Impressions* function, but

the difference is that *Clicks* only counts the impressions that led to a click.

$$\textit{Clicks}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \sum_{j \in [\eta_{ib}, \eta_{ie}]} \mathbb{1}(a_j \in \theta_{ia}) \mathbb{1}(p_j \in \theta_{ip}) \mathbb{1}(t_j \in \theta_{it}) \mathbb{1}(u_j \in \theta_{iu}) \mathbb{1}(y_j = 1) \tag{A.19}$$

Clicks is a good indicator of both ad and app performance. Moreover, at the user-level, the number of clicks captures an individual user's propensity to click, as well as her propensity to click within a specific ad and/or app. Thus, this is a very informative metric.

3. $\textit{CTR}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie})$: This function calculates the click-through rate (CTR) for a given set of inputs and history, *i.e.*, the ratio of clicks to impressions. If $\textit{Impressions}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) > 0$, then:

$$\textit{CTR}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \frac{\textit{Clicks}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie})}{\textit{Impressions}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie})}, \tag{A.20}$$

else if $\textit{Impressions}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = 0$, $\textit{CTR}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = 0$. The *CTR* function is a combination of *Impressions* and *Clicks*. It captures the click-propensity at the user, ad, app, and time-levels as well their interactions.[17]

4. $\textit{AdCount}(\theta_{ip}, \theta_{iu}; \eta_{ib}, \eta_{ie})$: This function returns the number of distinct ads shown for a given set of inputs.

$$\textit{AdCount}(\theta_{ip}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \sum_{a \in A^F} \mathbb{1}(\textit{Impressions}(\{a\}, \theta_{ip}, T, \theta_{iu}; \eta_{ib}, \eta_{ie}) > 0), \tag{A.21}$$

We use the full set of ads seen in our data (denoted by A^F) and not just the top 37 ads in this function.[18] This ensures that we are capturing the full extent of variety in ads seen by users. Features derived using this function capture the variety in ads for a given set of inputs. Past literature has argued that users are more likely to engage with ads when they are see a higher variety of ads (Redden, 2007). An example of a feature based on this function is $\textit{AdCount}(\{p_i\}, U; l, r_i)$, which counts the number of unique ads that were shown in app p_i across all users in the time period starting September 30 2015 and ending with the last

[17]In principle, we do not need to *CTR* over and above *Clicks* and *Impressions* if our learning model can automatically generate new features based on non-linear combinations of basic features. Machine learning models like Boosted Trees do this naturally and it is one of their big advantages. However, other methods like OLS and Logistic regressions cannot do automatic feature combination and selection. So we include this feature to help improve the performance of these simpler models. This ensures that we are not handicapping them too much in our model comparisons.

[18]We observe 263 unique ads in our data. Thus, the range of *AdCount* in our data goes from 0 to 263.

impression before i.

5. *Entropy*$(\theta_{ip}, \theta_{iu}; \eta_{ib}, \eta_{ie})$: This function captures the entropy or dispersion in the ads seen by a user. We use Simpson (1949)'s measure of diversity as our entropy metric.

$$\textit{Entropy}(\theta_{ip}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \frac{1}{\sum_{a \in A^F} \textit{Impressions}(\{a\}, \theta_{ip}, T, \theta_{iu}; \eta_{ib}, \eta_{ie})^2} \tag{A.22}$$

   Entropy contains information over and above just the count of the number of unique ads seen in the past since it captures the extent to which ads were dispersed across impressions. Consider two users who have both seen five impressions in the past, but with the following difference – (a) the first user has seen five distinct ads, or (b) the second has seen same ad five times. Naturally, the dispersion of ads is higher in case (a) than in case (b). The *Entropy* function reflects this dispersion – in case (a) the entropy is $\frac{1}{1^2+1^2+1^2+1^2+1^2} = 0.2$, whereas in case (b) it is $\frac{1}{5^2} = 0.04$. Thus, entropy is higher when ads are more dispersed.

   The literature on eye-tracking has shown that individuals' attention during advertisements reduces as ad-repetition increases (Pieters et al., 1999). We therefore expect the dispersion of previous ads to influence a user's attention span in our setting. Since attention is a prerequisite to clicking, we expect entropy-based measures to have explanatory power in our click prediction model.

6. *AppCount*$(\theta_{ia}, \theta_{iu}; \eta_{ib}, \eta_{ie})$: This function calculates the number of distinct apps in which a given ad is shown to a specific user. Similar to *AdCount*, it can be defined as follows:

$$\textit{AppCount}(\theta_{ia}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \sum_{p \in P} \mathbb{1}(\textit{Impressions}(\theta_{ia}, \{p\}, T, \theta_{iu}; \eta_{ib}, \eta_{ie}) > 0), \tag{A.23}$$

   where P is the set of apps, as defined in §4.3.1.[19] Previous studies have documented the spillover effects in multichannel online advertising (Li and Kannan, 2014). We therefore expect click probabilities to vary if a user saw an ad in just one app vs. saw it in many apps. We also expect the click behavior of users to vary based on the number of apps they use. *AppCount*-based features help us capture these differences.

7. *TimeVariability*$(\theta_{iu}; \eta_{ib}, \eta_{ie})$: This function measures the variance in a user's CTR over

[19] P has 51 elements, the top 1–50 apps and the 51^{st} element being the group of all smaller apps. In principle, we could use all the apps observed in the data and not just the Top 50 apps (like we did in the case of *AdCount*). However, we found that doing so does not really improve model performance. So we stick with this simpler definition.

different hours of the day. We can define this function as follows:

$$\textit{TimeVariability}(\theta_{iu}; \eta_{ib}, \eta_{ie}) = \mathrm{Var}_t[\textit{CTR}(A, P, \{t\}, \theta_{iu}; \eta_{ib}, \eta_{ie})] \tag{A.24}$$

Features based on this function capture variations in the temporal patterns in a user's clicking behavior, and we expect this information to help predict clicks.

8. *AppVariability*$(\theta_{iu}; \eta_{ib}, \eta_{ie})$: This function measures the variance in a user's CTR across different apps and is analogous to *TimeVariability*. This function is defined as follows:

$$\textit{AppVariability}(\theta_{iu}; \eta_{ib}, \eta_{ie}) = \mathrm{Var}_p[\textit{CTR}(A, \{p\}, T, \theta_{iu}; \eta_{ib}, \eta_{ie})] \tag{A.25}$$

   Note that both *TimeVariability* and *AppVariability* are defined at the user level.

We use the functions defined above to generate 98 features for each impression. In addition, we include a few standalone features such as a dummy for each of the top ads, the mobile and Internet service providers, latitude, longitude, and connectivity type as described below:

9. $Bid(a_i)$: The bid submitted for ad a_i shown in impression i.

10. $Latitude(u_i)$: The latitude of user u_i when impression i occurs.

11. $Longitude(u_i)$: The longitude of user u_i when impression i occurs.

12. $WiFi(u_i)$: Denotes whether the user is connected via Wi-Fi or mobile data plan during impression i. It can take two possible values, $\{0, 1\}$.

13. $Brand(u_i)$: Denotes the brand of the smartphone that user u_i is using. We observe eight smartphone brands in our data and therefore capture each brand using a dummy variable. So the range for this function is $\{0, 1\}^8$.

14. $MSP(u_i)$: Denotes the Mobile Service Provider used by user u_i during impression i. We observe four MSPs in our data. So the range for this function is $\{0, 1\}^4$.

15. $ISP(u_i)$: Denotes the Internet service providers (ISP) of user u_i. We observe nine unique ISPs in our data. So the range of this function is $\{0, 1\}^9$.

16. $AdDummy(a_i)$: Generates dummy variables for each of the top ads in the data. Although we expect our main features to capture many of the ad-fixed effects, we include ad dummies to capture any residual ad-fixed effects, e.g., banner design. The range of this function is $\{0, 1\}^{38}$, since there are 37 top ads and a 38th category comprising all the smaller ads.

B.3 Table of Features

In Table A1, we now present our full list of 160 features derived from the feature-functions described above as well as their classification as behavioral, contextual, and/or ad-specific based on the categorization scheme described in §4.3.3.

Table A1: List of Features for impression i.

No.	Feature Name	Feature Classification			Contextual Features	
		Behavioral	Contextual	Ad-specific	App-specific	Time-specific
1	*Impressions* $(A, P, T, u_i; l, r_i)$	✓				
2	*Impressions* $(a_i, P, T, U; l, g)$			✓		
3	*Impressions* $(A, p_i, T, U; l, g)$		✓		✓	
4	*Impressions* $(A, P, t_i, U; l, g)$		✓			✓
5	*Impressions* $(a_i, P, T, u_i; l, r_i)$	✓		✓		
6	*Impressions* $(A, p_i, T, u_i; l, r_i)$	✓	✓		✓	
7	*Impressions* $(A, P, t_i, u_i; l, r_i)$	✓	✓			✓
8	*Impressions* $(a_i, p_i, T, U; l, g)$		✓	✓	✓	
9	*Impressions* $(a_i, P, t_i, U; l, g)$		✓	✓		✓
10	*Impressions* $(A, p_i, t_i, U; l, g)$		✓		✓	✓
11	*Impressions* $(a_i, p_i, T, u_i; l, r_i)$	✓	✓	✓	✓	
12	*Impressions* $(a_i, p_i, t_i, U; l, g)$		✓	✓	✓	✓
13	*Impressions* $(A, P, T, u_i; s, r_i)$	✓				
14	*Impressions* $(a_i, P, T, U; s, g)$			✓		
15	*Impressions* $(A, p_i, T, U; s, g)$		✓		✓	
16	*Impressions* $(A, P, t_i, U; s, g)$		✓			✓
17	*Impressions* $(a_i, P, T, u_i; s, r_i)$	✓		✓		
18	*Impressions* $(A, p_i, T, u_i; s, r_i)$	✓	✓		✓	
19	*Impressions* $(A, P, t_i, u_i; s, r_i)$	✓	✓			✓
20	*Impressions* $(a_i, p_i, T, U; s, g)$		✓	✓	✓	
21	*Impressions* $(a_i, P, t_i, U; s, g)$		✓	✓		✓
22	*Impressions* $(A, p_i, t_i, U; s, g)$		✓		✓	✓
23	*Impressions* $(a_i, p_i, T, u_i; s, r_i)$	✓	✓	✓	✓	
24	*Impressions* $(a_i, p_i, t_i, U; s, g)$		✓	✓	✓	✓
25	*Impressions* $(A, P, T, u_i; o_i, r_i)$	✓				
26	*Impressions* $(a_i, P, T, u_i; o_i, r_i)$	✓		✓		
27	*Clicks* $(A, P, T, u_i; l, r_i)$	✓				
28	*Clicks* $(a_i, P, T, U; l, g)$			✓		
29	*Clicks* $(A, p_i, T, U; l, g)$		✓		✓	
30	*Clicks* $(A, P, t_i, U; l, g)$		✓			✓

Continued on next page

Table A1 – continued from previous page

No.	Feature Name	Feature Classification			Contextual Features	
		Behavioral	Contextual	Ad-specific	App-specific	Time-specific
31	*Clicks* $(a_i, P, T, u_i; l, r_i)$	✓		✓		
32	*Clicks* $(A, p_i, T, u_i; l, r_i)$	✓	✓		✓	
33	*Clicks* $(A, P, t_i, u_i; l, r_i)$	✓	✓			✓
34	*Clicks* $(a_i, p_i, T, U; l, g)$		✓	✓	✓	
35	*Clicks* $(a_i, P, t_i, U; l, g)$		✓	✓		✓
36	*Clicks* $(A, p_i, t_i, U; l, g)$		✓		✓	✓
37	*Clicks* $(a_i, p_i, T, u_i; l, r_i)$	✓	✓	✓	✓	
38	*Clicks* $(a_i, p_i, t_i, U; l, g)$		✓	✓	✓	✓
39	*Clicks* $(A, P, T, u_i; s, r_i)$	✓				
40	*Clicks* $(a_i, P, T, U; s, g)$			✓		
41	*Clicks* $(A, p_i, T, U; s, g)$		✓		✓	
42	*Clicks* $(A, P, t_i, U; s, g)$		✓			✓
43	*Clicks* $(a_i, P, T, u_i; s, r_i)$	✓		✓		
44	*Clicks* $(A, p_i, T, u_i; s, r_i)$	✓	✓		✓	
45	*Clicks* $(A, P, t_i, u_i; s, r_i)$	✓	✓			✓
46	*Clicks* $(a_i, p_i, T, U; s, g)$		✓	✓	✓	
47	*Clicks* $(a_i, P, t_i, U; s, g)$		✓	✓		✓
48	*Clicks* $(A, p_i, t_i, U; s, g)$		✓		✓	✓
49	*Clicks* $(a_i, p_i, T, u_i; s, r_i)$	✓	✓	✓	✓	
50	*Clicks* $(a_i, p_i, t_i, U; s, g)$		✓	✓	✓	✓
51	*CTR* $(A, P, T, u_i; l, r_i)$	✓				
52	*CTR* $(a_i, P, T, U; l, g)$			✓		
53	*CTR* $(A, p_i, T, U; l, g)$		✓		✓	
54	*CTR* $(A, P, t_i, U; l, g)$		✓			✓
55	*CTR* $(a_i, P, T, u_i; l, r_i)$	✓		✓		
56	*CTR* $(A, p_i, T, u_i; l, r_i)$	✓	✓		✓	
57	*CTR* $(A, P, t_i, u_i; l, r_i)$	✓	✓			✓
58	*CTR* $(a_i, p_i, T, U; l, g)$		✓	✓	✓	
59	*CTR* $(a_i, P, t_i, U; l, g)$		✓	✓		✓
60	*CTR* $(A, p_i, t_i, U; l, g)$		✓		✓	✓
61	*CTR* $(a_i, p_i, T, u_i; l, r_i)$	✓	✓	✓	✓	
62	*CTR* $(a_i, p_i, t_i, U; l, g)$		✓	✓	✓	✓
63	*CTR* $(A, P, T, u_i; s, r_i)$	✓				
64	*CTR* $(a_i, P, T, U; s, g)$			✓		
65	*CTR* $(A, p_i, T, U; s, g)$		✓		✓	
66	*CTR* $(A, P, t_i, U; s, g)$		✓			✓
67	*CTR* $(a_i, P, T, u_i; s, r_i)$	✓		✓		

Continued on next page

Table A1 – continued from previous page

No.	Feature Name	Feature Classification			Contextual Features	
		Behavioral	Contextual	Ad-specific	App-specific	Time-specific
68	*CTR* $(A, p_i, T, u_i; s, r_i)$	✓	✓		✓	
69	*CTR* $(A, P, t_i, u_i; s, r_i)$	✓	✓			✓
70	*CTR* $(a_i, p_i, T, U; s, g)$		✓	✓	✓	
71	*CTR* $(a_i, P, t_i, U; s, g)$		✓	✓		✓
72	*CTR* $(A, p_i, t_i, U; s, g)$		✓		✓	✓
73	*CTR* $(a_i, p_i, T, u_i; s, r_i)$	✓	✓	✓	✓	
74	*CTR* $(a_i, p_i, t_i, U; s, g)$		✓	✓	✓	✓
75	*AdCount* $(A, u_i; l, g)$	✓		✓		
76	*AdCount* $(p_i, U; l, g)$		✓	✓	✓	
77	*AdCount* $(p_i, u_i; l, g)$	✓	✓	✓	✓	
78	*AdCount* $(A, u_i; s, g)$	✓		✓		
79	*AdCount* $(p_i, U; s, g)$		✓	✓	✓	
80	*AdCount* $(p_i, u_i; s, g)$	✓	✓	✓	✓	
81	*AdCount* $(A, u_i; o_i, r_i)$	✓		✓		
82	*AppCount* $(A, u_i; l, g)$	✓	✓		✓	
83	*AppCount* $(a_i, U; l, g)$		✓	✓	✓	
84	*AppCount* $(a_i, u_i; l, g)$	✓	✓	✓	✓	
85	*AppCount* $(A, u_i; s, g)$	✓	✓		✓	
86	*AppCount* $(a_i, U; s, g)$		✓	✓	✓	
87	*AppCount* $(a_i, u_i; s, g)$	✓	✓	✓	✓	
88	*Entropy* $(A, u_i; l, g)$	✓		✓		
89	*Entropy* $(p_i, U; l, g)$		✓	✓	✓	
90	*Entropy* $(p_i, u_i; l, g)$	✓	✓	✓	✓	
91	*Entropy* $(A, u_i; s, g)$	✓		✓		
92	*Entropy* $(p_i, U; s, g)$		✓	✓	✓	
93	*Entropy* $(p_i, u_i; s, g)$	✓	✓	✓	✓	
94	*Entropy* $(A, u_i; o_i, r_i)$	✓		✓		
95	*TimeVariability* $(u_i; l, g)$	✓	✓			✓
96	*TimeVariability* $(u_i; s, g)$	✓	✓			✓
97	*AppVariability* $(u_i; l, g)$	✓	✓		✓	
98	*AppVariability* $(u_i; s, g)$	✓	✓		✓	
99	*Latitude* (u_i)	✓				
100	*Longitude* (u_i)	✓				
101	*WiFi* (u_i)	✓				
102-109	*Brand* (u_i)	✓				
110-113	*Operator* (u_i)	✓				
114-122	*ISP* (u_i)	✓				

Continued on next page

Table A1 – continued from previous page

No.	Feature Name	Feature Classification			Contextual Features	
		Behavioral	Contextual	Ad-specific	App-specific	Time-specific
123-160	*AdDummy* (a_i)			✓		

C XGBoost: Overview and Implementation

C.1 Overview of XGBoost

We start by considering a generic tree ensemble method as follows: let y_i and $\mathbf{x}_i$ denote the click indicator and the vector of features for impression i such that $y_i \in \{0, 1\}$ and $\mathbf{x}_i \in \mathbb{R}^k$, where k is the number of features. Then, a tree ensemble method is defined as follows:

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{j=1}^{J} \tau_j(\mathbf{x}_i) = \sum_{j=1}^{J} w^{(j)}_{q_j(\mathbf{x}_i)}, \tag{A.26}$$

where $q_j : \mathbb{R}^k \to \{1, 2, \dots, L_j\}$ and $w^{(j)} \in \mathbb{R}^{L_j}$ constitute the j^{th} regression tree τ_j with L_j leaves. Here q_j maps an impression to the leaf index and $w^{(j)}$ represents the weight on leaves. Hence, $w^{(j)}_{q_j(\mathbf{x}_i)}$ indicates the weight on the leaf that $\mathbf{x}_i$ belongs to. The tree ensemble method uses J additive functions to predict the output. In order to estimate the set of functions, Chen and Guestrin (2016) minimize the following regularized objective:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \gamma \sum_j L_j + \frac{1}{2}\lambda \sum_j \left\| w^{(j)} \right\|^2, \tag{A.27}$$

where γ and λ are the regularization parameters to penalize the model complexity, and l is a differentiable convex loss function (in our case, we use log loss as defined in § 5.1.1). Here, in contrast to MART, XGBoost penalizes not just tree depth but leaf weights as well.

Since the regularized objective in Equation (A.27) cannot be minimized using traditional optimization methods in Euclidean space, we employ Newton boosting in function space to train the model in an additive manner. Formally, if we define $\hat{y}_i^{(j)}$ as the prediction of the i^{th} impression at the j^{th} iteration, we will add τ_j to minimize the following objective:

$$\mathcal{L}(\phi) = \sum_i l\big(\hat{y}_i^{(j-1)} + \tau_j(\mathbf{x}_i), y_i\big) + \gamma L_j + \frac{1}{2}\lambda \left\| w^{(j)} \right\|^2 \tag{A.28}$$

In each iteration, we greedily add the tree that most improves our model according to the objective function in Equation (A.27). Since the model uses a greedy algorithm to find the best split at each iteration, it is impossible to search over all tree structures. Thus, we restrict the set of trees by specifying the maximum depth of a tree. To optimize this objective function, Friedman et al. (2000) propose a second-order approximation:

$$\mathcal{L}(\phi) \simeq \sum_i \left[l(\hat{y}_i^{(j-1)}, y_i) + \mathcal{G}_i \tau_j(\mathbf{x}_i) + \frac{1}{2}\mathcal{H}_i \tau_j^2(\mathbf{x}_i) \right] + \gamma L_j + \frac{1}{2}\lambda \left\| w^{(j)} \right\|^2, \tag{A.29}$$

where $\mathcal{G}_i = \partial_{\hat{y}_i^{(j-1)}} l(\hat{y}_i^{(j-1)}, y_i)$ and $\mathcal{H}_i = \partial^2_{\hat{y}_i^{(j-1)}} l(\hat{y}_i^{(j-1)}, y_i)$ are first and second order gradient statistics on the loss function. This approximation is used to derive the optimal tree at the step.

To implement this optimization routine, the researcher needs to provide a set of hyper-parameters. These include the regularization parameters γ and λ defined in Equation (A.27). Further, XGBoost uses two additional parameters to prevent over-fitting. The first is called shrinkage parameter (ν) introduced by Friedman (2002), which functions like learning rate in stochastic optimization. The second is the column sub-sampling parameter (χ_s), which is used to pick the fraction of features supplied to a tree and ranges from 0 to 1. In addition, we also need to specify parameters that structure the optimization problem and control the exit conditions for the algorithm. The first is d_{max}, the maximum depth of trees. As mentioned earlier, this ensures that the model searches over a finite set of trees. Second, we set the number of maximum number of iterations. Finally, the last parameter defines the early stopping rule, $\kappa \in \mathbb{Z}^+$, which stops the algorithm if the loss does not change in κ consecutive iterations. This parameter also helps prevent over-fitting.

Note all that these parameters cannot be inferred from the training data alone and should be set based on a scientific validation procedure. In the next section, §C.2, we provide a step by step explanation and the final values used for these hyper-parameters in our analysis.

C.2 Validation

The goal of validation is to pick the optimal tuning parameters. They cannot be inferred from the training data alone because they are hyper-parameters. The validation procedure uses two separate data-sets – training and validation data (from October 28 and 29, as shown in Figure 1) to pin down the hyper-parameters. It is worth noting that at this stage, the test data is kept separate and is brought out only at the end after the model has been finalized to evaluate the final model performance.

As discussed in §4.4, XGBoost uses five hyper-parameters that need tuning. Let $\mathcal{W} = \{\gamma, \lambda, \nu, d_{\max}, \chi_s\}$ denote the set of hyper-parameters, where γ and λ are the regularization parameters, ν is the shrinkage parameter or learning rate, $d_{\max}$ is maximum depth of trees, and χ_s is the column sub-sampling parameter, which refers to the percentage of features randomly selected in each round. For each of these parameters, we consider the following sets of values:

- $\gamma \in \{7, 9, 11\}$
- $\lambda \in \{0, 1, 2\}$
- $\nu \in \{0.05, 0.1, 0.5, 1\}$
- $d_{\max} \in \{5, 6, 7\}$
- $\chi_s \in \{0.5, 0.75\}$

Overall, $\mathcal{W}$ contains 216 elements. We now describe the validation and testing procedure in detail.

- Step 1: For each element of $\mathcal{W}$, train the model on the first two-thirds of the training and validation data and evaluate the model's performance on the remaining one-third. This is the typical hold-out procedure (Hastie et al., 2001). Boosted trees typically over-fit when they are stopped at some point. So when training the model, we use the early stopping rule to avoid this problem (Zhang et al., 2005).
- Step 2: Choose the set of hyper-parameters that gives the best model performance on the validation set (as measured by ***RIG*** in our case). Denote this as $\mathcal{W}^*$.
- Step 3: Using $\mathcal{W}^*$, train the final model on the full training and validation data, and here too we use the early stopping rule.
- Step 4: Evaluate the final model's performance on the test data.

Based on the above procedure, we derive the set of optimal hyper-parameters for our empirical setting as $\mathcal{W}^* = \{\gamma = 9, \lambda = 1, \nu = 0.1, d_{\max} = 6, \chi_s = 0.5\}$. Further, when training on the full training and validation data (Step 3), we do not see any improvement in model fit after 276 steps, so at this iteration number (following the early stopping rule).

Note that our validation procedure addresses two potential complications with our data and problem setting. First, our data and features have a time component. Thus, if we are not careful in how we split the validation and training data-sets, we can end up in situations where we use the future to predict the past (e.g., train on data from period t and validate on data from $t - 1$). To avoid this problem, the splits should be chosen based on time. Our validation procedure does this by using the first two-thirds of the validation+training data for training and the latter one-third for validation. However, doing so gives rise to a second problem – by choosing the most recent data for validation (instead of training), we forgo the information in the most recent impressions. In CTR prediction, it is important not to waste the most recent impressions while fitting the model because these impressions are more likely to be predictive of the future (McMahan et al., 2013). To address this, in Step 3, after choosing the optimal hyper-parameters, we train a final model on the full training and validation data. This model is then used as the final model for results and counterfactuals.

D Appendix for Robustness Checks

D.1 Other Evaluation Metrics

We consider three alternative evaluation metrics.

- Area Under the Curve (AUC): It calculates the area under the ROC curve, which is a graphical depiction of *true positive rate (TPR)* as a function of *false positive rate (FPR)*. This metric is often used in classification problems, but is less appropriate for prediction tasks such as ours because of two reasons. First, it is insensitive to the transformation of the predicted probabilities that preserve their rank. Thus, a poorly fitted model might have higher AUC than a well-fitted model (Hosmer Jr et al., 2013). Second, it puts the same weight on *false positive rate (FPR)* and *false negative rate (FNR)*. However, in CTR prediction, the penalty of FNR is usually higher than FPR (Yi et al., 2013).
- 0/1 Loss: This is a simple metric used to evaluate correctness in classification tasks. It is simply the percentage of incorrectly classified impressions. As with AUC, it is not very useful when accuracy of the prediction matters since it is not good at evaluating the predictive accuracy of rare events (e.g., clicks). For example, in our case, the loss is lower than 1% loss even if we blindly predict that none of the impressions will lead to click.
- Mean Squared Error (MSE): This is one of the most widely used metrics for measuring the goodness of fit. It is similar to LogLoss, which we use to calculate *RIG*. Both LogLoss and SquareLoss are often used for probability estimation and boosting in the machine learning literature. Let d_i be the Euclidean distance between the predicted value and actual outcome for impression i. This can be interpreted as the misprediction for the corresponding impression. SquareLoss and LogLoss for this impression will then be d_i^2 and $-\log(1-d_i)$ respectively. Since both functions are convex with respect to d_i, they penalize larger mispredictions more than smaller ones. However, a big difference is that SquareLoss is finite, whereas LogLoss is not. In fact, LogLoss evaluates $d_i = 1$ as infinitely bad. In our problem, this translates to either predicting 1 for non-clicks or predicting 0 for clicks. Therefore, the model optimized by LogLoss will not predict 0 or 1. Given that we do not know how users interact with the app at the moment, it is quite unrealistic to predict 1 for an impression, especially because each impression only lasts a short time. Thus, we choose LogLoss as our main metric, which is also the most common choice in the literature on CTR prediction (Yi et al., 2013).
- Confusion Matrix: We also present the Confusion Matrix for our main models. Like AUC and 0/1 Loss, Confusion Matrix is mostly used in classification problems. However, this gives us an

Data	Evaluation Metric	Behavioral	Contextual	Full
Full Sample	AUC	0.7910	0.7014	0.8230
Top Ads/Apps	AUC	0.8082	0.7192	0.8410
Filtered Sample	AUC	0.8073	0.7410	0.8410
Full Sample	0/1 Improvement (%)	0.63%	0.00%	4.74%
Top Ads/Apps	0/1 Improvement (%)	1.07%	0.00%	8.23%
Filtered Sample	0/1 Improvement (%)	1.34%	0.00%	8.47%
Full Sample	MSE Improvement (%)	3.41%	0.55%	8.59%
Top Ads/Apps	MSE Improvement (%)	4.86%	0.67%	13.33%
Filtered Sample	MSE Improvement (%)	4.89%	0.67%	12.93%
Full Sample	True Positives (#)	905	0	5934
Full Sample	True Negatives (#)	9533282	9533605	9531972
Full Sample	False Positives (#)	323	0	1633
Full Sample	False Negatives (#)	91325	92230	86296
Top Ads/Apps	True Positives (#)	757	0	5621
Top Ads/Apps	True Negatives (#)	6057514	6057744	6056235
Top Ads/Apps	False Positives (#)	230	0	1509
Top Ads/Apps	False Negatives (#)	50010	50767	45146
Filtered Sample	True Positives (#)	517	0	2928
Filtered Sample	True Negatives (#)	4424624	4424738	4424343
Filtered Sample	False Positives (#)	114	0	395
Filtered Sample	False Negatives (#)	29379	29896	26968
Full Sample	RIG	12.14%	5.25%	17.95%
Top Ads/Apps	RIG	14.82%	5.98%	22.85%
Filtered Sample	RIG	14.74%	6.77%	22.45%

Table A2: Model performance for the two samples (full and top ads/apps) when evaluated on the alternative metrics.

intuitive metric to evaluate the performance of our model.

We now take the optimized models presented in §5 and evaluate their performance on these alternative metrics. The results from this exercise are presented in Table A2. Overall, all our substantive results remain the same when we use alternative evaluation metrics.

D.2 Other Learning Methods

We now compare the performance of XGBoost with other five other learning algorithms and present the results in Table A3. Note that XGBoost outperforms all of them.

For each learning model, we describe the hyper-parameters associated with it and our approach to optimizing these hyper-parameters. Note that in all the cases, we use the same high-level validation procedure described in §C.2.

- Least squares does not use any hyper-parameters and hence does not require validation. In this case, we simply train the model on the full training and validation data to infer the model parameters, and report the model's performance on the test data.

Method	***RIG* over Baseline**
Least Squares	7.72%
LASSO	7.92%
Logistic Regression	11.17%
Regression Tree	15.03%
Random Forest	15.75%
XGBoost	17.95%

Table A3: *RIG* of different learning methods for the test data.

- For LASSO, the validation procedure is straightforward. The only hyper-parameter to set is the L1 regularization parameter, λ. We search over 100 values of λ and pick the one which gives us the best performance on the validation set ($\lambda = 6.3 \times 10^{-4}$). We then use this parameter and train the model on the entire training and validation set and test the performance on the test set.
- For CART, we use the package *rpart* in R, which implements a single tree proposed by (Breiman et al., 1984). We use recursive partitioning algorithm with a complexity parameter (c_{tree}) as the only hyper-parameter that we need to select through validation. The main role of this parameter is to avoid over-fitting and save computing time by pruning splits that are not worthwhile. As such, any split that does not improve the fit by a factor of complexity parameter is not attempted. We search for the optimal complexity parameter over the grid $c_{tree} \in \{0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001\}$, and based on the validation procedure, we derive the optimal complexity parameter as 5^{-5}. That is, the model adds another additional split only when the R-squared increments by at least 5^{-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) χ_s, the column sub-sampling parameter, which indicates the percentage of features that should be randomly considered in each round when looking for the best split, and (3) $n_{\min}$, the minimum number of samples required to split an internal node. We search for the optimal set of hyper-parameters over the following grid:
  - $n_{tree} \in \{100, 500, 1000\}$
  - $\chi_s \in \{0.33, 0.5, 0.75\}$
  - $n_{\min} \in \{100, 500, 1000\}$

  Based on our validation procedure, we find the optimal set of hyper-parameters to be: $\{n_{tree} = 1000, \chi_s = 0.33, n_{\min} = 500\}$.

D.3 Robustness Checks on Feature Generation

We also ran the following robustness checks on the feature generation framework.

- We start by considering different ways of aggregating over the history. One possibility is to use $\eta_{ie} = r_i$ for all the features, i.e., update all the features in real-time. We find no difference in terms of the prediction accuracy when we adopt this approach, though it increases the computational costs significantly. Therefore, we stick to our current approach where we use a combination of global and real-time features.
- Next, we examine the model's performance under an alternative definition of long- and short-term history for features that are updated in real-time. The idea is to fix the length of history, instead of fixing η_{ib} for each impression. In other words, instead of aggregating over $[l, r_i]$ and $[s, r_i]$ where l and s are fixed, we aggregate over $[l_i, r_i]$ and $[s_i, r_i]$ where l_i and s_i are no more fixed, but the length of $[l_i, r_i]$ and $[s_i, r_i]$ are fixed. For example, l_i for impression i on October 28 is the same time on September 30, while l_i for impression i on October 30 is the same time on October 2. Under this new approach, we find a slight decrease in the performance: the *RIG* drops to 17.69% improvement over the baseline.
- We also consider a model with dummy variables for the top apps in our feature set (similar to what we now do for top ads). Again, we find no discernible differences in the results without app dummies: the *RIG* is 17.97% over the baseline. This may be due to the fact that our feature generation framework captures the fixed effects of apps well.

Overall, we find that our feature set works well and any additional features or more complex feature generation mechanisms do not provide any significant benefits in *RIG*.

D.4 Sampling and Data Adequacy

We conduct our analyses using a relatively large sample consisting of 727,354 users in the train, validation, and test data-sets. This corresponds to 17,856,610 impressions in the training and validation data, and 9,625,835 impressions in the test data. We now examine the adequacy the rate the adequacy of our sample by calculating the RIG for different (lower) sample sizes. That is, we quantify how much our model gains by using more data, and at what point the marginal value of additional data is minimal.

To calculate the *RIG* for a given sample size of N_u, we do the following: 1) We take ten random samples of N_u users, and generate two data sets – the training data and the test data. 2) For each sample, we train the model using the training data and then test the model's performance on the test data.[20] 3) We then calculate the mean and standard deviation of the *RIG* for each sample. We

[20]We use the hyper-parameters obtained from the validation exercise that we performed in the main model for training.

User Sample Size (N_u)	$\overline{N}_{train}$	$\overline{N}_{test}$	**RIG over Baseline CTR**	
			Coefficient	Std. error
1,000	24,500	13,274	13.76%	3.17%
5,000	124,521	66,820	14.25%	1.76%
10,000	249,749	139,123	15.26%	1.48%
20,000	486,007	266,497	16.14%	0.40%
50,000	1,220,394	663,569	16.84%	0.28%
100,000	2,436,037	1,332,894	17.27%	0.23%
200,000	4,875,586	2,654,110	17.58%	0.20%
400,000	9,749,402	5,327,471	17.84%	0.18%
600,000	14,699,589	7,928,275	17.91%	0.15%

Table A4: RIG for different sample sizes. $\overline{N}_{train}$ and $\overline{N}_{test}$ are respectively the average size of train and test data after sampling users.

perform this exercise for nine sample sizes starting with $N_u = 1000$ and going up till $N_u = 600,000$. The results from this exercise are shown in Table A4. We also report the average sample size of train and test data respectively as $\overline{N}_{train}$ and $\overline{N}_{test}$.

In principle, we can perform the above exercise for each sample size with only one sample instead of ten. However, such an approach is likely to be error-prone, especially at smaller sample sizes, since there is heterogeneity among users and each sample is random. So we may randomly find a smaller sample to have a higher *RIG* than a larger sample in one particular instance. To avoid making incorrect inferences due to the particularities of one specific sample and to minimize the noise in our results, we employ the bootstrap procedure described above.

Table A4 suggests that after about 100,000 users, increasing the sample size improves the prediction only slightly. However, increasing sample sizes also increase the training time and computational costs. Given the cost-benefit trade-off, our sample of 727,354 users is more than sufficient for our purposes.

D.5 Other Validation Techniques

The validation procedure outlined in Appendix §C.2 is the first-best validation procedure in data-rich situations such as ours. Nevertheless, we examine two other commonly used techniques:

- Hold-out validation – very similar to our current approach, except that at Step 3, instead of training the final model on the combination of training and validation data, we simply use the best model (using the optimal $\mathcal{W}^*$) trained based on the training data (from Step 2) as the final

This is likely to help the performance of the models trained on smaller samples because if we were to tune the model using smaller data, the estimated hyper-parameters are likely to be worse. Thus, the gains reported here more favorable than what we would obtain if we also validated/tuned the model using the smaller data samples.

model. Thus, we do not use the validation data to train the final model. This can lead to some information loss (especially from the recent impressions). We find that the model performance on the test set drops when we use hold-out validation: our *RIG* is 17.21% which is lower than that of our validation procedure.

- k-fold cross-validation – we find no improvement in the performance of the model selected by 5-fold cross-validation (*RIG* is 17.33%). Please see Footnote 7 in (Yoganarasimhan, 2019) and (Hastie et al., 2001) for a detailed discussion on the pros and cons of k-fold cross validation.

E Detailed Analysis of the Example Presented in Figure 8

In an important paper, Levin and Milgrom (2010) argue that sharing too much targeting information with advertisers can thin auction markets which in turn would soften competition and make the platform worse. We now present a simple example to highlight this idea. Consider a platform with two advertisers ($a^{(1)}$ and $a^{(2)}$) competing for the impressions of two users ($u^{(1)}$ and $u^{(2)}$). Assume that the platform uses second price auctions with Cost per Impression (CPI) pricing, where the highest bidder wins the impression and pays the bid of the second-highest bidder. These auctions have the useful property of truthful bidding (Vickrey, 1961). Let the advertisers be symmetric in their valuation of a click (normalized to 1 hereafter). Further, let the match values between advertisers and users be as shown in Equation (A.30). Match values can be interpreted as the eCTR of an impression for the advertiser-user pair. Notice that advertiser $a^{(1)}$ has a better match with user $u^{(1)}$ and advertiser $a^{(2)}$ with user $u^{(2)}$.

$$eCTR = \begin{array}{c c} & \begin{array}{cc} u^{(1)} & u^{(2)} \end{array} \\ \begin{array}{c} a^{(1)} \\ a^{(2)} \end{array} & \begin{pmatrix} 0.5 & 0.1 \\ 0.1 & 0.3 \end{pmatrix} \end{array} \longrightarrow \begin{array}{c} \bar{u} \\ \begin{pmatrix} 0.3 \\ 0.2 \end{pmatrix} \end{array} \tag{A.30}$$

We can formally show that, for a given targeting strategy, both CPI and Cost per Click (CPC mechanisms generate the same revenues for the platform and advertisers. So we focus on the CPI case throughout the text and refer readers to Appendix G for the CPC example and analysis.

Consider the advertiser's bidding strategy and outcomes under two regimes – 1) No data disclosure by the platform, and 2) Full disclosure of match values by the platform. The results from these two regimes are laid out in Table A5 and discussed below:

- No data disclosure – Here advertisers only know their aggregate match value for both users. So $a^{(1)}$ and $a^{(2)}$'s expected match values for the two users are 0.3 and 0.2. In a second price auction, advertisers bid their expected valuations. So $a^{(1)}$ wins both impressions and pays the next

No Data Disclosure (or No Targeting)	Full Data Disclosure (or Perfect Targeting)
For both impressions: Bids: Advertiser $a^{(1)}$: $b_1^{(1)} = b_2^{(1)} = 0.3$ Advertiser $a^{(2)}$: $b_1^{(2)} = b_2^{(2)} = 0.2$ Outcome: Advertiser $a^{(1)}$ wins both impressions and pays 0.2 per impression	For User $u^{(1)}$'s impression: Bids: Advertiser $a^{(1)}$: $b_1^{(1)} = 0.5$, Advertiser $a^{(2)}$: $b_1^{(2)} = 0.1$ Outcome: Advertiser $a^{(1)}$ wins $u^{(1)}$'s impression and pays 0.1 For User $u^{(2)}$'s impression: Bids: Advertiser $a^{(1)}$: $b_2^{(1)} = 0.1$, Advertiser $a^{(2)}$: $b_2^{(2)} = 0.3$ Outcome: Advertiser $a^{(2)}$ wins $u^{(2)}$'s impression and pays 0.1
Platform's expected revenue: $R = 2 \times 0.2 = 0.4$ Advertiser's expected surplus: $W^{(1)} = 2 \times (0.3 - 0.2) = 0.2$ $W^{(2)} = 0$ Total expected surplus: $S = 0.6$	Platform's expected revenue: $R = 0.1 + 0.1 = 0.2$ Advertiser's expected surplus: $W^{(1)} = 0.5 - 0.1 = 0.4$ $W^{(2)} = 0.3 - 0.1 = 0.2$ Total expected surplus: $S = 0.8$

Table A5: Example depicting two regimes: 1) No data disclosure and 2) Full disclosure.

highest bid, $b_1^{(2)} = b_2^{(2)} = 0.2$, for each impression. Therefore, platform's revenue is $R = 0.4$, advertisers' surplus is $W^{(1)} = 0.2$ and $W^{(2)} = 0$, and the total surplus is $S = 0.6$.

- Full data disclosure – Since advertisers now have information on their match for each impression, they submit targeted bids that reflect their valuations as shown in Table A5. Therefore, the advertiser who values the impression more wins it. However, because of the asymmetry in advertisers' valuation over impressions, the competition over each impression is softer. This ensures higher advertiser revenues, with $W^{(1)} = 0.4$ and $W^{(2)} = 0.2$. However, the platform's revenue is now lower, at $R = 0.2$. Thus, even though ads are matched more efficiently and the total surplus generated is higher, the platform extracts less revenue.

This example illustrates the platform's trade-off between value creation and value appropriation, and highlights the platform's incentives to withhold targeting information from advertisers.

F Proof of Proposition 1

We first prove the proposition for the general case of two sets of bundles $\mathcal{I}^{(1)}$ and $\mathcal{I}^{(2)}$, where $\mathcal{I}^{(1)}$ is at least as granular as $\mathcal{I}^{(2)}$. Under a given targeting level, risk-neutral advertisers' valuation for an impression in a bundle is their aggregate valuation for that bundle, as they cannot distinguish between different impressions within a bundle. Thus, all impressions within a bundle have the same expected valuation for an advertiser. As such, for any $i \in I_j$, incentive compatibility constraint in the second-price auction induce advertiser a to places the bid $b_{ia} = \frac{1}{|I_j|} \sum_{k \in I_j} v_{ka}$.

According to Definition 4, for any $I_k^{(2)} \in \mathcal{I}^{(2)}$, there exist $I_{j_1}^{(1)}, \dots, I_{j_l}^{(1)}$ such that $\bigcup_{s=1}^{l} I_{j_s}^{(1)} = I_k^{(2)}$. We can write:

$$\max_a \sum_{i \in I_k^{(2)}} v_{ia} \leq \sum_{s=1}^{l} \max_a \sum_{i \in I_{j_s}^{(1)}} v_{ia} \tag{A.31}$$

where the LHS is the surplus generated from the impressions in bundle $I_k^{(2)}$ for targeting level $\mathcal{I}^{(2)}$, and RHS is the surplus generated from the same impressions for targeting level $\mathcal{I}^{(1)}$. Since inequality (A.31) holds for any $I_k^{(2)} \in \mathcal{I}^{(2)}$, we can show that $S^{(1)} \geq S^{(2)}$, *i.e.*, the surplus is increasing with more granular targeting.

However, platform revenues can go either way. If the second-highest valuation is exactly the same as the highest valuation for all impressions, the revenue and surplus are identical, i.e., $R^{(1)} \geq R^{(2)}$. On the other hand, consider a case where, for any given impression, one advertiser has a high valuation and the rest of advertisers have the same (lower) valuation. In this case, the second-highest valuation is the minimum valuation for each impression. Thus, we can write (A.31) with $\min$ function instead of $\max$ and change the sign of inequality. This gives us $R^{(1)} \leq R^{(2)}$. Therefore, the platform revenues can be non-monotonic with more granular targeting.

Finally, it is worth noting that this result can be applied to any efficient auction in light of revenue-equivalence theorem.

G Analysis of Cost-per-Click Payment Mechanism

We now present an analysis of targeting under the Cost-per-Click (CPC) mechanism, where an advertiser's bid indicates the maximum price he is willing to pay per click. In this case, having a more accurate estimation of match values for impressions does not change advertisers' bidding behavior because they do not pay per impression. Theoretically, if advertisers have infinite budget, they should bid their valuation for click, even if they know they have higher CTR for some impressions.

However, the ad-network has an incentive to make more efficient matches in order to generate more clicks since clicks are their main source of revenue. For example, if there is an advertiser with a very high bid but very low CTR, the ad-network cannot make money by selling the slot to this advertisers. Let v_{ia} and m_{ia} respectively denote the valuation for click and the match value for advertiser a for impression i. The maximum revenue that platform could get is then

No Targeting	Perfect Targeting
For both impressions: Bids: Advertiser 1: $b_{11} = b_{21} = 1$ Advertiser 2: $b_{12} = b_{22} = 1$ Match values: Advertiser 1: $m_{11} = m_{21} = 0.3$ Advertiser 2: $m_{12} = m_{22} = 0.2$ Outcomes: Advertiser 1 wins both impressions and pays $\frac{0.2 \times 1}{0.3} = 0.66$ per click	For User 1's impression: Bids: Advertiser 1: $b_{11} = 1$, Advertiser 2: $b_{12} = 1$ Match values: Advertiser 1: $m_{11} = 0.5$, Advertiser 2: $m_{12} = 0.1$ Outcome: Advertiser 1 wins User 1's impression and pays $\frac{1 \times 0.1}{0.5} = 0.2$ per click For User 2's impression: Bids: Advertiser 1: $b_{11} = 1$, Advertiser 2: $b_{12} = 1$ Match values: Advertiser 1: $m_{11} = 0.1$, Advertiser 2: $m_{12} = 0.3$ Outcome: Advertiser 2 wins User 2's impression and pays $\frac{1 \times 0.1}{0.3} = 0.33$ per click
Platform's expected revenue: $R = 0.66 \times (0.5 + 0.1) = 0.4$ Advertiser's expected surplus: $W_1 = (1 - 0.66) \times (0.5 + 0.1) = 0.2$ $W_2 = 0$ Total expected surplus: $S = 0.6$	Platform's expected revenue: $R = 0.2 \times 0.5 + 0.33 \times 0.3 = 0.2$ Advertiser's expected surplus: $W_1 = (1 - 0.2) \times 0.5 = 0.4$ $W_2 = (1 - 0.33) \times 0.3 = 0.2$ Total expected surplus: $S = 0.8$

Table A6: Example depicting no targeting and perfect targeting under CPC pricing mechanism.

$\max_a v_{ia} m_{ia}$. Thus, defining b_{ia} as advertiser a's bid on impression i, the platform should sell the ad slot to $\text{argmax}_a \, b_{ia} m_{ia}$ and charge her the minimum bid with which she still wins. It generates the expected revenue of the second-highest $b_{ia} m_{ia}$. (In fact, this is how Google's sponsored search auctions work.)

Table A6 shows how the example in §6.1 generalizes to the CPC case. Although CPC and CPI have different properties, we can easily prove that their revenue is the same under different levels of targeting. This stems from the fact that under second-price auction, bidders bid their valuation. Thus, the platform's expected revenue from impression i under both mechanisms is the second highest $v_{ia} m_{ia}$, as long as the match-values (or targeting strategies) are the same under both mechanisms. Conceptually, there exists a one-to-one mapping between CPC and CPI mechanisms if and only if there is a one-to-one mapping between the targeting level and resulting match-value matrix. In the CPI mechanism, m_{ia} enters the advertisers' bidding strategy, i.e., the targeting decision is made by the advertisers. The ad-network's decision consists of whether to share targeting information with advertisers or not. In contrast, under the CPC mechanism, the ad-network directly decides the extent of targeting to engage in and the advertisers always bid their valuation.

Our empirical results suggest that under the CPI mechanism, ad-networks may have incentive to limit behavioral targeting. In the CPC context, this translates to the following result: the ad-network has an incentive to not use behavioral information for targeting, as compared to contextual information. In both cases, the platform has an incentive to protect users' privacy by ensuring that behavioral information is not used for targeting purposes. In sum, the empirical estimates of platform's revenue, advertisers' revenues, and the implications for user-privacy are similar in both settings.

H Robustness Checks for Analysis of Revenue-Efficiency Trade-off

H.1 Alternative Methods for Estimation of Click Valuations

Here we discuss alternative methods to estimate click valuations from the data. For the main model, we use a simple method that approximates the click valuation as twice the submitted bid. This comes from the following relationship established in Mirrokni et al. (2010):

$$\hat{v}_a^{(c)} = b_a^* + \frac{b_a^*}{1-\pi_a}, \tag{A.32}$$

where π_a is the probability of winning for ad a in the auction. Since this probability is very small for advertisers, we neglect that in our approximation. Further, in our problems, ads are charged using a next-price mechanism like that of Google's sponsored search auctions. In this case, the amount that ads are charged per click is not their bid, but rather the amount that guarantees their rank among the bidders. For example, if there are three bidders with bids 1, 2, and 3, and quality scores 0.1, 0.2, and 0.3, the scores will be 0.1, 0.4, and 0.9 respectively. Now, if the second-ranked bidder gets a click, she will pay the price that would have guaranteed her rank. That is, she only needs to pay $\frac{1\times 0.1}{0.2} = 0.5$, as it guarantees her score to be higher than the third-ranked bidder. Together, not incorporating either π_a in (A.32) or the cost function would make our estimates biased. To address this issue, we consider alternative methods to estimate click valuations as presented below:

1. **Alternative Method 1 (AM1):** We only incorporate the proportions as specified in (A.32). We use a simple approach to estimate proportions as follows:

$$\hat{\pi}_a = \frac{\sum_{i=1}^{N_F} \mathbb{1}(a_i = a)}{\sum_{i=1}^{N_F} e_{i,a}} \tag{A.33}$$

This is the total number of impressions showing ad a divided by the total number of impressions ad a participated in their auction. This is a consistent estimator for the proportion of impressions ad a would get, given her bid. Thus, in this method, we can estimate valuations as follows:

$$\hat{v}_a^{(AM1)} = b_a^* + \frac{b_a^*}{1 - \hat{\pi}_a}, \tag{A.34}$$

2. **Alternative Method 2 (AM2):** Not only do we incorporate proportions, we also incorporate the cost function. It is easy to show that if the cost function $c_a(b_a)$ is differentiable and concave, the game is *socially concave* and there exists a unique pure strategy Nash equilibrium (Rosen, 1965; Mirrokni et al., 2010). We can write the following relationship for the equilibrium bid:

$$\hat{v}_a^{(c)} = c_a(b_a^*) + \frac{b_a^* c'_a(b_a^*)}{1 - \pi_a}, \tag{A.35}$$

where c'_a is the first derivative of the cost function. In this method, we assume a linear form for function cost function such that $c_a(b_a) = \gamma_a b_a$. Our estimate for click valuations will then be:

$$\hat{v}_a^{(AM2)} = \hat{c}_a(b_a^*) + \frac{\hat{c}_a(b_a^*)}{1 - \hat{\pi}_a}, \tag{A.36}$$

where we can estimate $\hat{c}_a(b_a^*)$ by taking the average cost-per-click of ad a.

3. **Alternative Method 3 (AM3):** In this method, we follow (A.35) but we use a quadratic form for the cost function such that $c_a(b_a) = \gamma_a b_a + \gamma b_a^2$, where the coefficient for b_a is ad specific but the one for b_a^2 is the same for all ads. The main reason is identification since advertisers do not change their bids. So the curvature is captured by the data of all ads. With a quadratic form for the cost function, we can re-write (A.35) as follows:

$$\hat{v}_a^{(c)} = c_a(b_a^*) + \frac{b_a^*(\gamma_a + 2\gamma b_a^*)}{1 - \pi_a} = c_a(b_a^*) + \frac{b_a^* c_a(b_a^*)}{1 - \pi_a} + \frac{\gamma b_a^*}{1 - \pi_a} \tag{A.37}$$

Now, to estimate γ, we need to estimate the following model:

$$\mathit{CPC}_i = \hat{\gamma}_a b_a + \hat{\gamma} b_a^2 + \hat{\epsilon}, \tag{A.38}$$

where CPC_i is the actual cost-per-click for impression i. We then plug these estimates and in (A.35). Our estimates in this case will be:

$$\hat{v}_a^{(AM3)} = \hat{c}_a(b_a^*) + \frac{\hat{c}_a(b_a^*)}{1 - \hat{\pi}_a} + \frac{\hat{\gamma} b_a^*}{1 - \hat{\pi}_a}, \tag{A.39}$$

where $\hat{c}_a(b_a^*)$ and $\hat{\pi}_a$ can be estimated using the approach in AM2.

4. **Alternative Method 4 (AM4):** In this method, we incorporate the fact that there is a reserve price r_0 for all impressions that requires advertisers to bid higher than this amount. In presence of reserve price, equilibrium bids may not satisfy first-order condition, as they are right-censored. That is, there could have been bids lower than r_0 with no reserve price. For reserve bidders, we know that the participation constraint $v_a^{(c)} \geq r_0$ is satisfied. On the other hand, we know that $b_a^* \leq r_0$ which implies that $v_a^{(c)} \leq r_0(1+\frac{1}{1-\pi_a})$. Thus, for reserve bidders, we have $v_a^c \in [r_0, r_0(1+\frac{1}{1-\pi_a})]$. As such, if such valuations come from a uniform distribution in this range, our estimates will be:

$$\hat{v}_a^{(AM4)} = \begin{cases} r_0(1+\frac{1}{2(1-\pi_a)}) & \text{if } a \text{ is a reserve bidder} \\ \hat{v}_a^{(AM3)} & \text{otherwise} \end{cases}, \tag{A.40}$$

where the estimated click valuations for a reserve bidder a is the mean of uniform distribution $\mathcal{U}(r_0, r_0(1+\frac{1}{1-\pi_a}))$.

5. **Alternative Method 5 (AM5):** In this method, we follow all the steps in AM4, but draw a value from distribution $\mathcal{U}(r_0, r_0(1+\frac{1}{1-\pi_a}))$ for any ad a who bids the reserve price. As such, our estimates will be:

$$\hat{v}_a^{(AM4)} = \begin{cases} x \sim \mathcal{U}(r_0, r_0(1+\frac{1}{1-\pi_a})) & \text{if } a \text{ is a reserve bidder} \\ \hat{v}_a^{(AM3)} & \text{otherwise} \end{cases} \tag{A.41}$$

We first illustrate the empirical CDF of all these alternative methods as well as the main method used in the paper. As shown in Figure 11, estimated value distributions are not very different. This is mostly due to the fact that there are many bidders and no one has disproportionately high chance of winning, which in turn, makes the approximation method used in (12) valid.

Next, we use all the alternative methods to obtain click valuation estimates and then estimate the market outcomes – total surplus, platform revenues, and advertisers' surplus. The results are shown in Table A7. While our estimates are quantitatively different from those presented in Table 6, our main qualitative results remain the same: while total surplus has a monotonic relationship with the granularity of targeting, platform revenues are maximized when the platform limits the targeting level to the contextual targeting.

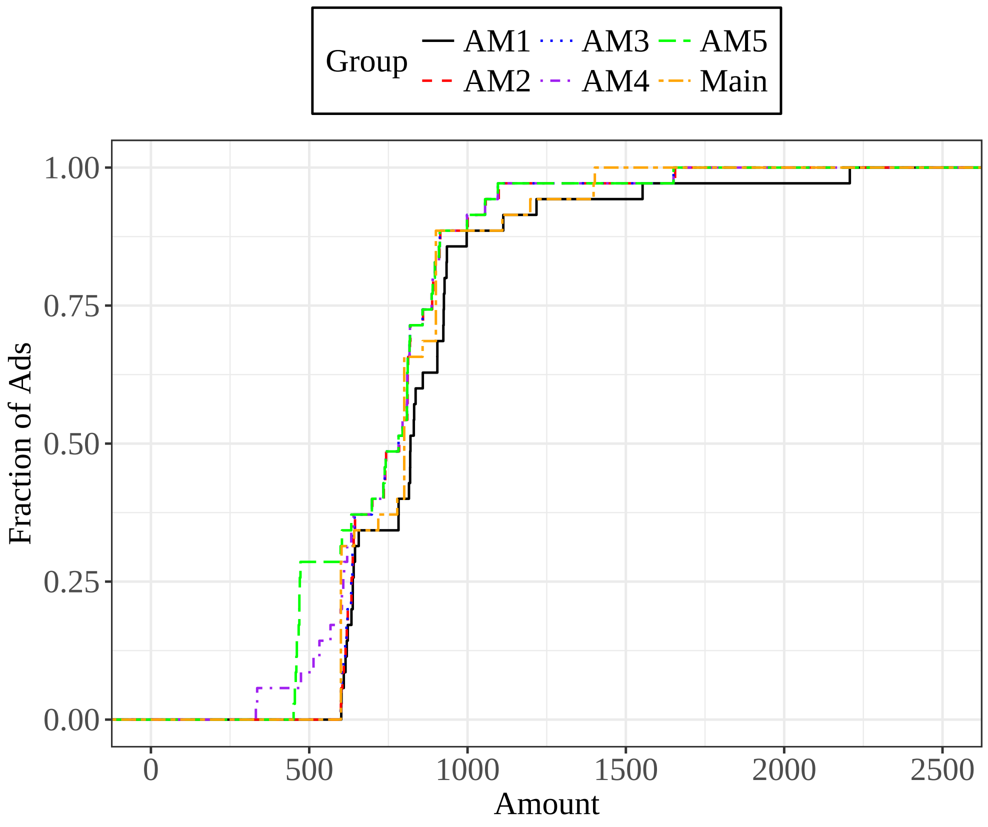


Figure 11: Empirical CDF of estimated value distributions using alternative methods

H.2 Adding Noise to Match Valuations

As discussed in §6.3.2, our assumption that advertisers can obtain match values estimated in our machine learning framework may not be realistic. As such, we consider various scenarios here to reflect cases in which advertisers can only obtain a noisy version of our estimates.

H.2.1 Identically Distributed Noise Across Ads

First, we consider the case that an identically distributed noise is added to any element in the match value matrix. We operationalize this noise as follows:

$$\hat{m}_{i,a}^{(\nu)} = \hat{m}_{i,a}\epsilon_{i,a}, \tag{A.42}$$

where $\epsilon_{i,a} \sim \mathcal{U}(1-\nu, 1+\nu)$. As such, advertisers obtain estimates $\nu \times 100$ percent lower or higher than the estimate from our ML framework. For any noise distribution, we make the entire matrix and follow the procedure presented in §6.3.3 to estimate the average surplus, revenue, and advertisers' surplus. The results are shown in Table A8. We find that noise-adding can increase the

Targeting	Total Surplus	Platform Revenue	Advertisers' Surplus
A. Alternative Method 1 (AM1)			
No Targeting	8.54	8.36	0.18
Contextual Targeting	9.09	8.49	0.60
Behavioral Targeting	9.32	8.47	0.85
Full Targeting	9.60	8.48	1.12
B. Alternative Method 2 (AM2)			
No Targeting	7.96	7.93	0.03
Contextual Targeting	8.62	8.09	0.53
Behavioral Targeting	8.83	8.05	0.78
Full Targeting	9.09	8.05	1.04
C. Alternative Method 3 (AM3)			
No Targeting	7.91	7.89	0.02
Contextual Targeting	8.61	8.07	0.53
Behavioral Targeting	9.81	8.04	0.77
Full Targeting	9.08	8.04	1.04
D. Alternative Method 4 (AM4)			
No Targeting	8.02	7.91	0.11
Contextual Targeting	8.61	8.08	0.53
Behavioral Targeting	8.82	8.03	0.79
Full Targeting	9.08	8.03	1.05
E. Alternative Method 5 (AM5)			
No Targeting	8.02	7.91	0.11
Contextual Targeting	8.61	8.08	0.53
Behavioral Targeting	8.82	8.03	0.79
Full Targeting	9.08	8.04	1.04

Table A7: Platform revenues, advertisers' surplus, and total surplus for different levels of targeting using different methods to estimate click valuations. The numbers are reported in terms of the average monetary unit per impression.

revenue under full targeting by distorting efficiency. This is in line with Athey and Nekipelov (2010) who find that the platform's optimal decision is to use imperfect CTR estimates in a cost-per-click setting. Interestingly, we find that full and contextual targeting yield almost the same results when $\nu = 0.3$. Thus, we can interpret contextual targeting as a noise-adding strategy.

H.2.2 Differentially Distributed Noise Across Ads

Here we consider a more realistic case wherein ads with more impressions (more data) have better estimates (less noisy). Similar to the case with identical noise, we we operationalize the differential noise as follows for any ad a:

$$\hat{m}_{i,a}^{(\nu_a)} = \hat{m}_{i,a}\epsilon_{i,a}, \tag{A.43}$$

Targeting	**Noise (ν)**	**Total Surplus**	**Platform Revenue**	**Advertisers' Surplus**
No Targeting	0.05	8.36	8.30	0.06
Contextual Targeting	0.05	8.99	8.44	0.55
Behavioral Targeting	0.05	9.18	8.35	0.82
Full Targeting	0.05	9.43	8.35	1.08
No Targeting	0.1	8.36	8.30	0.06
Contextual Targeting	0.1	8.99	8.44	0.55
Behavioral Targeting	0.1	9.17	8.36	0.82
Full Targeting	0.1	9.37	8.35	1.02
No Targeting	0.15	8.36	8.30	0.06
Contextual Targeting	0.15	8.99	8.44	0.55
Behavioral Targeting	0.15	9.16	8.36	0.80
Full Targeting	0.15	9.29	8.35	0.93
No Targeting	0.2	8.36	8.30	0.06
Contextual Targeting	0.2	8.99	8.44	0.55
Behavioral Targeting	0.2	9.14	8.36	0.78
Full Targeting	0.2	9.20	8.37	0.82
No Targeting	0.25	8.36	8.30	0.06
Contextual Targeting	0.25	8.99	8.44	0.55
Behavioral Targeting	0.25	9.13	8.37	0.76
Full Targeting	0.25	9.10	8.41	0.70
No Targeting	0.3	8.36	8.30	0.06
Contextual Targeting	0.3	8.99	8.44	0.54
Behavioral Targeting	0.3	9.12	8.38	0.74
Full Targeting	0.3	9.01	8.47	0.54

Table A8: Platform revenues, advertisers' surplus, and total surplus for different levels of targeting when adding an identically distributed noise to match values. The numbers are reported in terms of the average monetary unit per impression.

where $\epsilon_{i,a} \sim \mathcal{U}(1-\nu_a, 1+\nu_a)$. As such, the noise is different for each ad. Since the error rate is proportional to the square root of number of observations, we write:

$$\nu_a = \frac{\nu}{\sqrt{N_a}},$$

where N_a is the number of impressions ad a has in our Filtered sample. Now, for any ν, we can make the entire match value matrix and estimate the market outcomes. The results are shown in Table A9. Again, we find that the platform benefits when advertisers have imperfect estimates of their match valuations. Comparing the results in Table A8 and A9, we find that the total surplus declines faster with a differentially distributed noise: when the platform's revenue is the same under full and contextual targeting, the total surplus is higher under contextual targeting. Given the absence of privacy costs under contextual targeting, our results in both tables indicate that platforms benefit most when allowing only contextual targeting.

Targeting	**Noise (ν)**	**Total Surplus**	**Platform Revenue**	**Advertisers' Surplus**
No Targeting	10	8.36	8.30	0.06
Contextual Targeting	10	8.99	8.44	0.55
Behavioral Targeting	10	9.18	8.35	0.82
Full Targeting	10	9.42	8.34	1.08
No Targeting	20	8.36	8.30	0.06
Contextual Targeting	20	8.99	8.44	0.55
Behavioral Targeting	20	9.17	8.35	0.82
Full Targeting	20	9.35	8.32	1.03
No Targeting	30	8.36	8.30	0.06
Contextual Targeting	30	8.99	8.44	0.55
Behavioral Targeting	30	9.16	8.35	0.80
Full Targeting	30	9.24	8.31	0.93
No Targeting	40	8.36	8.30	0.06
Contextual Targeting	40	8.99	8.44	0.55
Behavioral Targeting	40	9.14	8.35	0.79
Full Targeting	40	9.12	8.32	0.80
No Targeting	50	8.36	8.30	0.06
Contextual Targeting	50	8.99	8.44	0.55
Behavioral Targeting	50	9.12	8.35	0.77
Full Targeting	50	8.99	8.35	0.65
No Targeting	60	8.36	8.30	0.06
Contextual Targeting	60	8.99	8.44	0.55
Behavioral Targeting	60	9.10	8.35	0.75
Full Targeting	60	8.93	8.36	0.57
No Targeting	70	8.36	8.30	0.06
Contextual Targeting	70	8.99	8.43	0.56
Behavioral Targeting	70	9.09	8.34	0.75
Full Targeting	70	8.86	8.39	0.47
No Targeting	80	8.36	8.30	0.06
Contextual Targeting	80	8.99	8.43	0.55
Behavioral Targeting	80	9.07	8.34	0.73
Full Targeting	80	8.80	8.44	0.36

Table A9: Platform revenues, advertisers' surplus, and total surplus for different levels of targeting when adding an differentially distributed noise to match values. The numbers are reported in terms of the average monetary unit per impression.

References

S. Athey and D. Nekipelov. A structural model of sponsored search advertising auctions. In *Sixth Ad Auctions Workshop*, volume 15, 2010.

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

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, R. Tibshirani, et al. Additive Logistic Regression: A Statistical View of Boosting (with Discussion and a Rejoinder by the Authors). *The Annals of Statistics*, 28(2):337–407, 2000.

J. H. Friedman. Stochastic Gradient Boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning. *NY Springer*, 2001.

D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant. *Applied Logistic Regression*, volume 398. John Wiley & Sons, 2013.

G. A. Johnson, R. A. Lewis, and D. Reiley. Location, Location, Location: Repetition and Proximity Increase Advertising Effectiveness. *Available at SSRN 2268215*, 2016.

J. Levin and P. Milgrom. Online Advertising: Heterogeneity and Conflation in Market Design. *The American Economic Review*, 100(2):603–607, 2010.

H. Li and P. Kannan. Attributing Conversions in a Multichannel Online Marketing Environment: An Empirical Model and a Field Experiment. *Journal of Marketing Research*, 51(1):40–56, 2014.

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

V. Mirrokni, S. Muthukrishnan, and U. Nadav. Quasi-Proportional Mechanisms: Prior-Free Revenue Maximization. In *Latin American Symposium on Theoretical Informatics*, pages 565–576. Springer, 2010.

R. Pieters, E. Rosbergen, and M. Wedel. Visual Attention to Repeated Print Advertising: A Test of Scanpath Theory. *Journal of Marketing Research*, pages 424–438, 1999.

J. P. Redden. Reducing Satiation: The Role of Categorization Level. *Journal of Consumer Research*, 34(5): 624–634, 2007.

J. B. Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica: Journal of the Econometric Society*, pages 520–534, 1965.

N. S. Sahni. Effect of temporal spacing between advertising exposures: Evidence from online field experiments. *Quantitative Marketing and Economics*, 13(3):203–247, 2015.

E. H. Simpson. Measurement of Diversity. *Nature*, 1949.

W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of finance*, 16(1): 8–37, 1961.

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. *Forthcoming, Management Science*, 2019.

T. Zhang, B. Yu, et al. Boosting with Early Stopping: Convergence and Consistency. *The Annals of Statistics*, 33(4):1538–1579, 2005.