Web Appendix for "Targeting and Privacy in Mobile Advertising"

Omid Rafieian* University of Washington

Hema Yoganarasimhan* University of Washington

April 23, 2020

A Propensity Scores and Unconfoundedness Assumption

Our empirical strategy relies on randomization in the allocation of ads, which gives us the unconfoundedness assumption and allows us to perform targeting counterfactuals. As outlined in §4.2.2 in the main text of the paper, one diagnostic test for unconfoundedness assumption is to assess covariate balance. For this, we first need to estimate propensity scores, and then examine whether adjusting for these propensity scores achieves covariate balance. We do the former in §A.1 and the latter in §A.2.

A.1 Estimation of Propensity Scores

We now propose our approach to estimate propensity scores. First, recall the allocation rule presented in Equation (1) in the main text of the paper:

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

Our goal is to estimate π_{ia} for all top 37 ads in each impression i. In our filtering procedure, we identify ads with zero propensity of being shown and filter them out of the competition. However, our filtering procedure does not estimate the magnitude of the propensity score for ads with non-zero propensity scores. We therefore discuss our procedure to estimate propensity scores below.

The outcome and covariates for our propensity score model are listed below:

*Please address all correspondence to: rafieian@uw.edu, hemay@uw.edu.

1. *Outcome*: The main outcome of interest is the ad shown. This is a categorical variable that has 38 classes, i.e., $a_i \in A = \{a^{(1)}, a^{(2)}, \ldots, a^{(37)}, a^{(s)}\}$, where $a^{(1)}, a^{(2)}, \ldots, a^{(37)}$ refer to the identities of the top 37 ads, and all the smaller ads are grouped into one category denoted by $a^{(s)}$.

2. *Covariates*: The set of covariates that we control for are:

   - Filtering indicators for each ad $e_{i,a}$. This controls for whether the ad a is in $\mathcal{A}_i$.
   - All the targeting variables, including province, app category, hour of the day, smartphone brand, connectivity type, and MSP.
   - Exact location (latitude and longitude), to control for any geographical targeting beyond province-level targeting.
   - Minute-level time of the day, to control for any shocks or changes in the quality scores and bids in the system. Note that we do not expect bids or quality scores to change within the observation window. Nevertheless, this control simply ensures that there are no biases even if they did due to system bugs etc.

We use a multi-class XGBoost model to regress our outcome of interest on the set of covariates. We use multi-class logarithmic loss as the evaluation metric, which is defined as follows:

$$\mathcal{L}^{mlog\ loss}(\hat{\pi}, \mathbf{a}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j\in A}\mathbb{1}(a_i = a^{(j)})\log(\hat{\pi}_{i,j}), \tag{A.1}$$

where the summation over j refers to all the outcome classes that we have. Using a softmax objective, we can then estimate the propensity scores. Figure A.1 shows the distribution of propensity scores for top eight ads, using box plots. As shown in this figure, the average propensity of winning for each ad is below 0.25. Further, there is no disproportionately high propensity of winning: the highest is slightly above 0.50. We also notice that Ad 5 has higher median propensity score than Ad 2, even though the total number of impressions is higher for Ad 2. This is because Ad 5 is targeting more narrowly, and is not available for all the impressions.

A.2 Assessing Covariate Balance Using Inverse Probability Weighting

Now, we use our estimated propensity scores to assess covariate balance across different ads. We begin by describing the balance for a simple case in which we have a fully randomized experiment with one treatment and one control group in §A.2.1. We then extend the same notion to the case where we want to assess balance using propensity scores across multiple treatment arms §A.2.2.

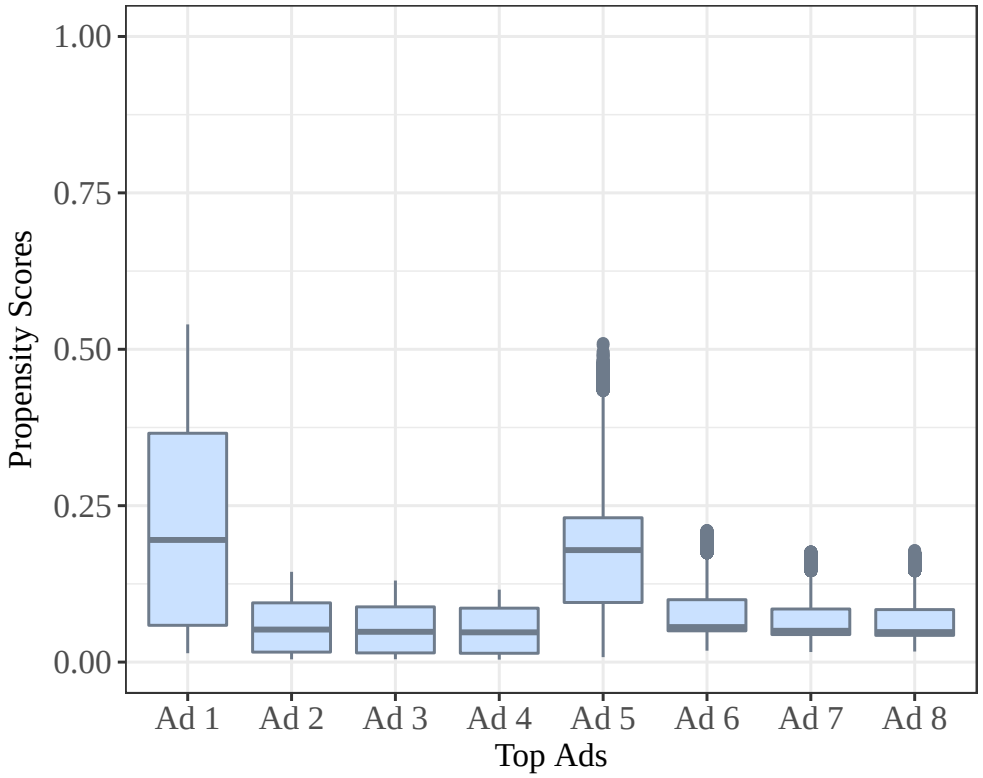

Figure A.1: Distribution of propensity scores for top 8 ads

A.2.1 Covariate Balance in a Fully Randomized Experiment with a Binary Treatment

To assess the covariate balance in a fully randomized experiment with two treatment arms, we simply need to show that the distribution of covariate Z is the same across treated and control samples. As such, we rely on standardized bias, which is the absolute mean difference in covariate Z across the treatment and control groups, divided by the standard deviation of the Z in the pooled sample:

$$SB(Z) = \frac{|\bar{Z}_{\text{treated}} - \bar{Z}_{\text{control}}|}{\sigma_Z}, \tag{A.2}$$

where $\bar{Z}_{\text{treated}}$ and $\bar{Z}_{\text{control}}$ are the mean value of covariate Z for the treated and control groups, and σ_Z is the standard deviation of Z for the pooled sample.

Clearly, the lower the standardized bias is, we have greater balance in covariate Z across treatment and control groups. In practice, different cutoffs are considered acceptable by the researchers. A common cutoff is 0.20 – that is, a standardized bias greater than 0.20 is taken as evidence for imbalance (McCaffrey et al., 2013). In this paper, we follow this norm and set our cutoff to be 0.20.

A.2.2 Covariate Balance Using Inverse Propensity Weights for Multiple Treatments

Our empirical setting is more complicated than a fully randomized experiment with a binary treatment because of two reasons: (1) ad assignment is not fully exogenous, but a function of observed covariates, so covariate balance may not be satisfied for the unweighted distribution of covariates, and (2) unlike a binary treatment, we have multiple treatment arms (ads), so the balance is not simply the difference between two groups. To address the first issue, we need to use our estimated propensity scores to weight our original sample such that an observation with a lower propensity score gets a larger weight to represent what the sample would have been if the platform had run a fully randomized experiment. For the second issue, we follow the approach in McCaffrey et al. (2013) and compare the weighted mean of covariate Z when assigned to ad with the population mean of that covariate and check for the balance by measuring the standardized bias. In principle, if weights are accurate, the weighted mean of covariate Z when assigned to ad a must be the same as the population mean. As such, we define the weight-adjusted standardized bias for covariate Z when assigned to ad a as follows:

$$SB_a^{\hat{\pi}}(Z) = \frac{|\bar{Z}_a^{\hat{\pi}} - \bar{Z}|}{\sigma_Z}, \tag{A.3}$$

where superscript $\hat{\pi}$ indicates the propensity score estimates used to weight the samples, and $\bar{Z}_a^{\hat{\pi}}$ is defined as follows:

$$\bar{Z}_a^{\hat{\pi}} = \frac{\sum_{i=1}^{N} \frac{\mathbb{1}(a_i=a)}{\hat{\pi}_{ia}} Z_i}{\sum_{i=1}^{N} \frac{\mathbb{1}(a_i=a)}{\hat{\pi}_{ia}}}, \tag{A.4}$$

where Z_i is the value of covariate Z in impression i, the term $\mathbb{1}(a_i = a)$ indicates whether ad a is actually shown in that impression, and the weights are simply the inverse propensity scores.

We can then calculate the weight-adjusted standardized bias measure for each ad a and define the standardized bias for variable Z as follows:

$$SB^{\hat{\pi}}(Z) = \max_a SB_a^{\hat{\pi}}(Z), \tag{A.5}$$

where by construction, if $SB^{\hat{\pi}}(Z)$ satisfies the balance criteria, it means that each $SB_a^{\hat{\pi}}(Z)$ satisfies this criteria. To have a baseline of how imbalanced covariates were before adjusting for inverse propensity weights, we define the unweighted measure of standardized bias for covariate Z when assigned to ad a as follows:

$$SB(Z) = \max_a SB_a(Z) = \max_a \frac{|\bar{Z}_a - \bar{Z}|}{\sigma_Z}, \tag{A.6}$$

Comparing $SB(Z)$ (unweighted standardized bias) and $SB^{\hat{\pi}}(Z)$ (weight-adjusted standardized bias) allows us to see how adjusting for inverse propensity scores helped balance covariates.

Now, we focus on all the targeting variables in our sample and calculate both unweighted and weight-adjusted measures of standardized bias. Targeting variables are the main candidates that could potentially fail the balance test because advertisers can target these variables. Since all these variables are categorical, for each targeting variable (Z), we need to assess balance for the dummy variable for each separate category within that targeting variable.

For a given targeting variable (e.g., province), let $\mathcal{Z}$ denote the set of dummy variables corresponding to each sub-category within that targeting variable. We define the standardized bias as the maximum standardized bias across all its sub-categories. As such, for both unweighted and weight-adjusted methods, we have:

$$SB^{\hat{\pi}}(\mathcal{Z}) = \max_{Z \in \mathcal{Z}} SB^{\hat{\pi}}(Z) = \max_{Z \in \mathcal{Z}} \max_{a} SB_a^{\hat{\pi}}(Z)$$
$$SB(\mathcal{Z}) = \max_{Z \in \mathcal{Z}} SB(Z) = \max_{Z \in \mathcal{Z}} \max_{a} SB_a(Z)$$

Figure A.2 shows both unweighted and weight-adjusted measures of standardized bias for all of our targeting variables (on the filtered test data sample, which is used for the counterfactual analysis). Notice that all six targeting variables are initially imbalanced in the actual data. In particular, province, hour of the day, and MSP are the variables with the largest magnitude of imbalance. However, after adjusting for inverse propensity scores, we find that all measures of standardized bias are below the 0.2 cutoff. Thus, we have covariate balance.

What is striking about Figure A.2 is the very large measures of unweighted standardized bias for some targeting variables. This is due to the presence of some ads with very low propensity scores and sparse categories within certain targeting variables. For example, there are some provinces in our data that constitute less than 1% of our observations. Likewise, there are some ads with less than 1% share of observations. As a result, there are very few observations where these ads are shown in these provinces. This may result in substantial imbalance in the original data.

Next, we focus only on ads that are shown at least in 1% of our sample and make the same balance plot in Figure A.3. In this case, the measures of standardized bias are significantly lower compared to Figure A.2, especially the unweighted measures. Interestingly, we find that four out of six covariates are balanced in the unweighted sample: app category, connectivity type, MSP, and smartphone brand. This is because these variables are not widely used for targeting and their sub-categories have a sufficiently large number of observations. The only two variables that are imbalanced in the original sample are state and hour of the day, most likely because these are the

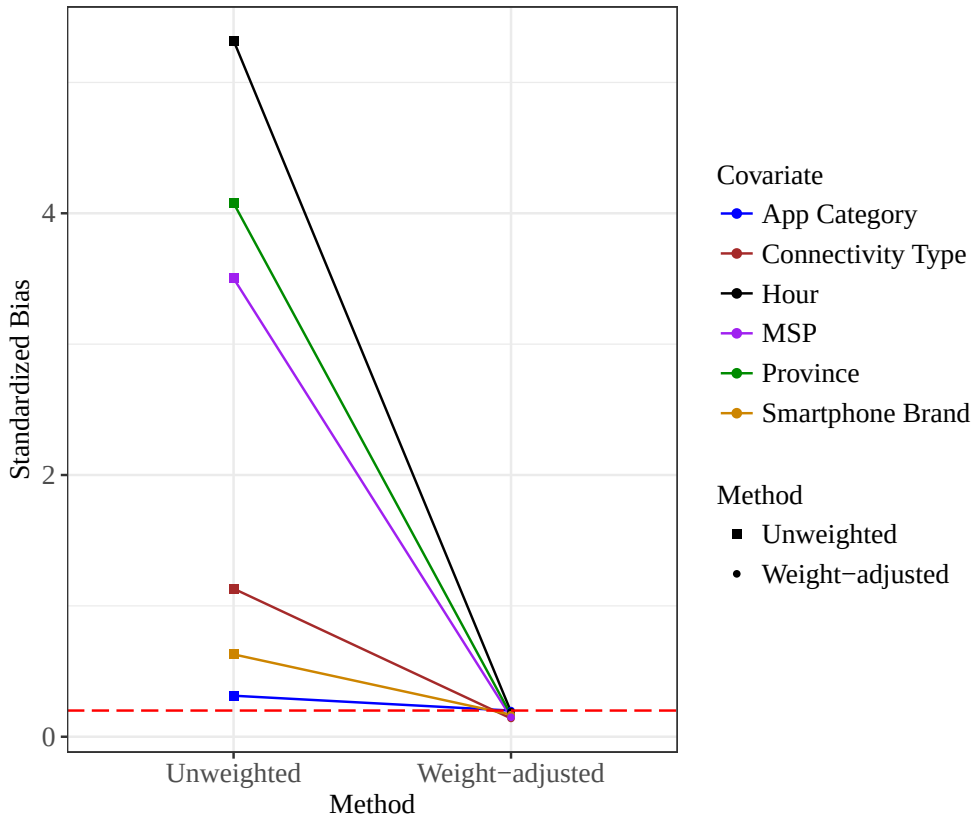

Figure A.2: Weight-adjusted and unweighted measures of standardized bias for all targeting variables (balance plot). The dotted red line shows the 0.2 cut-off.

variables are most used for targeting among advertisers. We also find that in both cases, hour of the day has the highest standardized bias (even though we achieve balance at the 0.2 cutoff in both cases). This may also be due to the unavailability of some ads because of reasons other than targeting such as budget exhaustion.

In sum, the above analysis presents strong empirical evidence for covariate balance in our data after adjusting for propensity scores. This suggests the unconfoundedness assumption is valid in our setting.

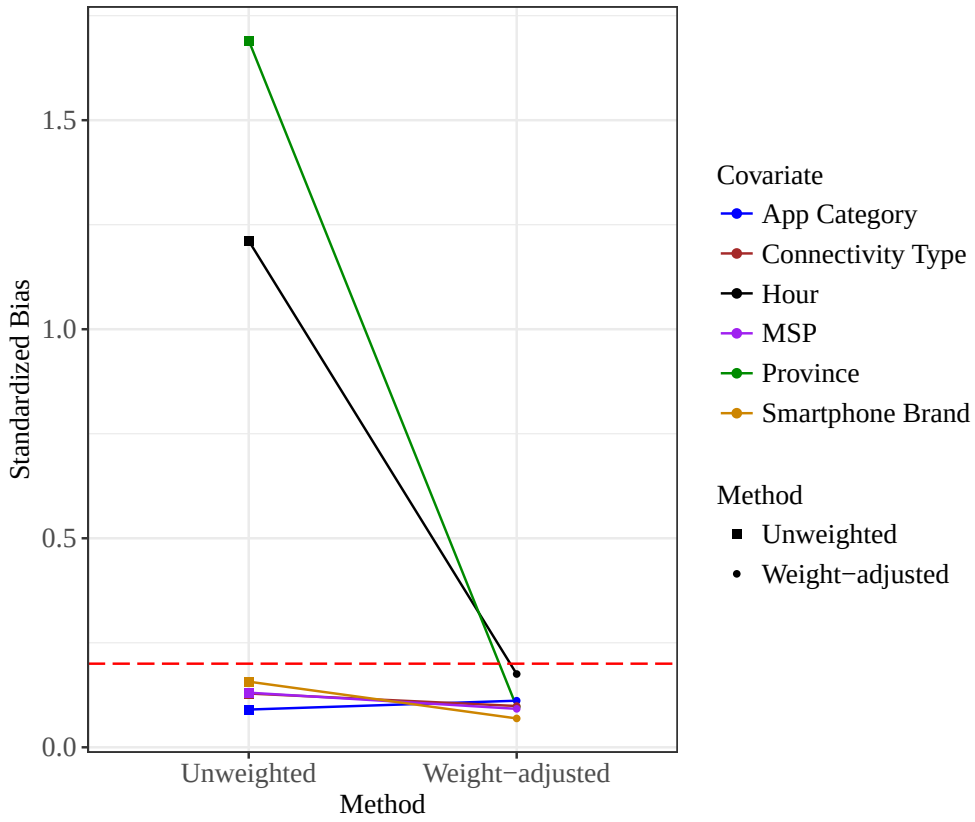

Figure A.3: Weight-adjusted and unweighted measures of standardized bias for all targeting variables, when we only consider ads with over 1% share of total number of observations in the filtered sample in test data (balance plot). The dotted red line shows the 0.2 cut-off.

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 (2020). 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 $a_i \in A = \{a^{(1)}, a^{(2)}, \ldots, a^{(37)}, a^{(s)}\}$. The

elements $a^{(1)}, a^{(2)}, \ldots, a^{(37)}$ 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 $p_i \in P = \{p^{(1)}, p^{(2)}, \ldots, p^{(50)}, p^{(s)}\}$. $p^{(1)}$ 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: $T = \{1, 2, \ldots, 24\}$.
- the user u_i who generated the impression, where $u_i \in U = \{u^{(1)}, \ldots, u^{(728,340)}\}$. 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.[1] 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 A.4, 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.

$$\textit{Impressions}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \sum_{j \in [\eta_{ib}, \eta_{ie}]} \mathbb{1}(a_j \in \theta_{ia}) \mathbb{1}(p_j \in \theta_{ip}) \mathbb{1}(t_j \in \theta_{it}) \mathbb{1}(u_j \in \theta_{iu}), \tag{A.7}$$

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}.

[1] 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.

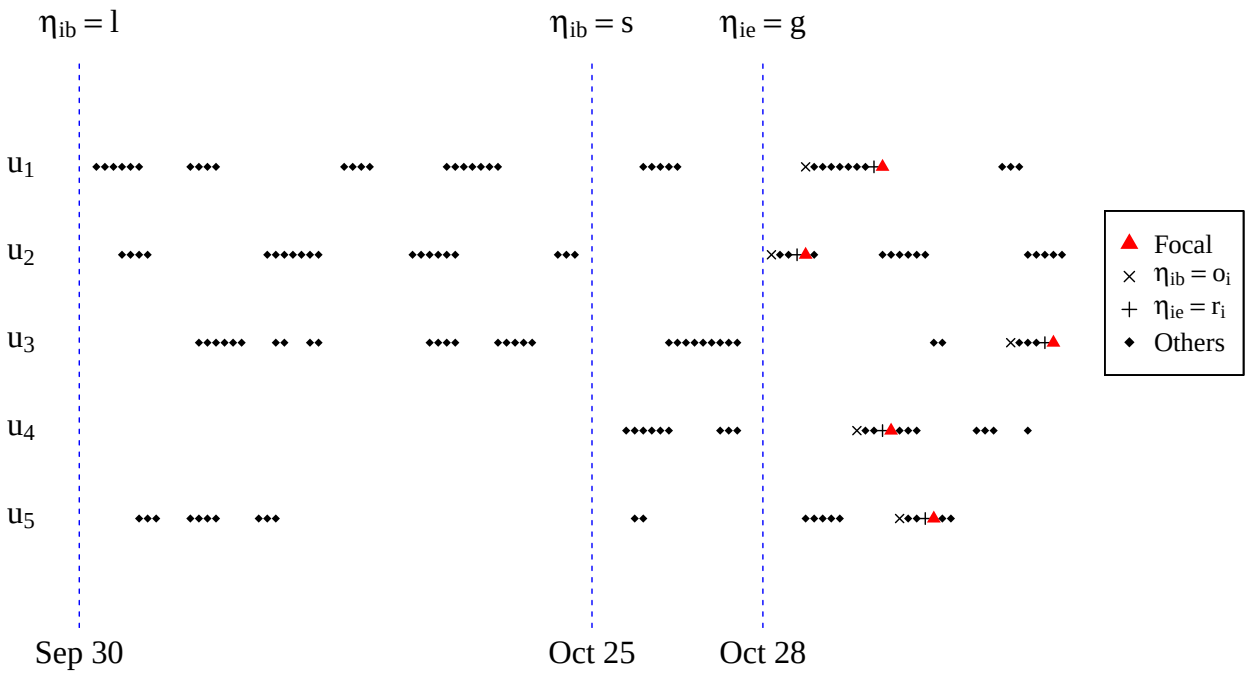

Figure A.4: Depiction of history for five example users.The ▲ refers to the focal impression i for which the features are being generated. $+$ denotes the last impression just before the focal impression, and $\times$ refers to the first impression in the session in which the focal impression occurs.

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 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.

$$\textit{Clicks}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \sum_{j \in [\eta_{ib}, \eta_{ie}]} \mathbb{1}(a_j \in \theta_{ia}) \mathbb{1}(p_j \in \theta_{ip}) \mathbb{1}(t_j \in \theta_{it}) \mathbb{1}(u_j \in \theta_{iu}) \mathbb{1}(y_j = 1) \tag{A.8}$$

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. *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 *Impressions*$(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) > 0$, then:

$$\textit{CTR}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \frac{\textit{Clicks}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie})}{\textit{Impressions}(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie})}, \tag{A.9}$$

else if *Impressions*$(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = 0$, *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.[2]

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

$$\textit{AdCount}(\theta_{ip}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \sum_{a \in A^F} \mathbb{1}(\textit{Impressions}(\{a\}, \theta_{ip}, T, \theta_{iu}; \eta_{ib}, \eta_{ie}) > 0), \tag{A.10}$$

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.[3] 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. In a recent paper, Rafieian and Yoganarasimhan (2020) show that higher variety of previous ads increases the CTR on the next ad.

[2] 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.

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

An example of a feature based on this function is *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 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.

$$\textit{Entropy}(\theta_{ip}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \frac{1}{\sum_{a \in A^F} \textit{Impressions}(\{a\}, \theta_{ip}, T, \theta_{iu}; \eta_{ib}, \eta_{ie})^2} \tag{A.11}$$

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:

$$\textit{AppCount}(\theta_{ia}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \sum_{p \in P} \mathbb{1}(\textit{Impressions}(\theta_{ia}, \{p\}, T, \theta_{iu}; \eta_{ib}, \eta_{ie}) > 0), \tag{A.12}$$

where P is the set of apps, as defined in §B.1.[4] 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.

[4] 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.

7. *TimeVariability*$(\theta_{iu}; \eta_{ib}, \eta_{ie})$: This function measures the variance in a user's CTR over different hours of the day. We can define this function as follows:

$$\textit{TimeVariability}(\theta_{iu}; \eta_{ib}, \eta_{ie}) = \mathrm{Var}_t[\textit{CTR}(A, P, \{t\}, \theta_{iu}; \eta_{ib}, \eta_{ie})] \tag{A.13}$$

   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:

$$\textit{AppVariability}(\theta_{iu}; \eta_{ib}, \eta_{ie}) = \mathrm{Var}_p[\textit{CTR}(A, \{p\}, T, \theta_{iu}; \eta_{ib}, \eta_{ie})] \tag{A.14}$$

   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 38^{th} 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 in the main text of the paper.

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* $(A, P, T, u_i; l, r_i)$	✓				

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
28	*Clicks* $(a_i, P, T, U; l, g)$			✓		
29	*Clicks* $(A, p_i, T, U; l, g)$		✓		✓	
30	*Clicks* $(A, P, t_i, U; l, g)$		✓			✓
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* $(a_i, P, T, U; s, g)$			✓		

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
65	*CTR* $(A, p_i, T, U; s, g)$		✓		✓	
66	*CTR* $(A, P, t_i, U; s, g)$		✓			✓
67	*CTR* $(a_i, P, T, u_i; s, r_i)$	✓		✓		
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)	✓				

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
102-109	*Brand* (u_i)	✓				
110-113	*Operator* (u_i)	✓				
114-122	*ISP* (u_i)	✓				
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:

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{j=1}^{J} \tau_j(\mathbf{x}_i) = \sum_{j=1}^{J} w^{(j)}_{q_j(\mathbf{x}_i)}, \tag{A.15}$$

where $q_j : \mathbb{R}^k \to \{1, 2, \ldots, L_j\}$ 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:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \gamma \sum_j L_j + \frac{1}{2}\lambda \sum_j \left\| w^{(j)} \right\|^2, \tag{A.16}$$

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 in the main text of the paper). Here, in contrast to MART, XGBoost penalizes not just tree depth but leaf weights as well.

Since the regularized objective in Equation (A.16) 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:

$$\mathcal{L}(\phi) = \sum_i l\big(\hat{y}_i^{(j-1)} + \tau_j(\mathbf{x}_i), y_i\big) + \gamma L_j + \frac{1}{2}\lambda \left\|w^{(j)}\right\|^2 \tag{A.17}$$

In each iteration, we greedily add the tree that most improves our model according to the objective function in Equation (A.16). 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:

$$\mathcal{L}(\phi) \simeq \sum_i \left[l(\hat{y}_i^{(j-1)}, y_i) + \mathcal{G}_i\tau_j(\mathbf{x}_i) + \frac{1}{2}\mathcal{H}_i\tau_j^2(\mathbf{x}_i)\right] + \gamma L_j + \frac{1}{2}\lambda \left\|w^{(j)}\right\|^2, \tag{A.18}$$

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.16). 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 of the Web Appendix, §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 in the main

text of the paper) 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 earlier in §C.1, 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 Evaluating Efficient Targeting Policy Using Importance Sampling

As discussed in §5.2 in the main text of the paper, the overlap between our data and the efficient targeting policy provides a unique opportunity to use model-free approaches to evaluate the efficient targeting policy. Intuitively, if we weight the overlapping sample such that it represents the full sample, we can estimate the outcome of interest under the efficient targeting policy. In particular, we use *importance sampling* whereby we weight the impressions in the overlapping area by the inverse propensity score of those impressions. We can now define the average match value under the efficient targeting as follows:

$$\hat{m}_{IS}^{\tau^*} = \frac{1}{N_F} \sum_{i=1}^{N_F} \sum_{a \in \mathcal{A}_i} \frac{\mathbb{1}(a_i = a)\mathbb{1}(\tau^*(i) = a)}{\hat{\pi}_{ia}} y_{i,a}, \tag{A.19}$$

where $y_{i,a}$ is the click outcome for ad a in impression i, and the indicator functions both take value one only if the efficient targeting policy coincides the actual data. As such, for non-zero elements in the summation above, we actually observe the actual outcome of interest $y_{i,a}$. Now, we can even show that under unconfoundedness assumption, $\hat{m}_{IS}^{\tau^*}$ is a consistent estimator for the match value under the efficient targeting policy, conditional on observed covariates. We can write:

$$
\begin{aligned}
\mathbb{E}\left[\hat{m}_{IS}^{\tau^*} \mid X\right] &= \mathbb{E}\left[\frac{1}{N_F}\sum_{i=1}^{N_F}\sum_{a\in\mathcal{A}_i}\frac{\mathbb{1}(a_i=a)\mathbb{1}(\tau^*(i)=a)}{\hat{\pi}_{ia}}y_{i,a} \mid X\right] \\
&= \frac{1}{N_F}\sum_{i=1}^{N_F}\sum_{a\in\mathcal{A}_i}\mathbb{E}\left[\frac{\mathbb{1}(a_i=a)\mathbb{1}(\tau^*(i)=a)}{\hat{\pi}_{ia}}y_{i,a} \mid X\right] \\
&= \frac{1}{N_F}\sum_{i=1}^{N_F}\sum_{a\in\mathcal{A}_i}\frac{\hat{\pi}_{ia}\mathbb{1}(\tau^*(i)=a)}{\hat{\pi}_{ia}}\mathbb{E}\left[y_{i,a} \mid X\right] \\
&= \frac{1}{N_F}\sum_{i=1}^{N_F}\sum_{a\in\mathcal{A}_i}\mathbb{1}(\tau^*(i)=a)\mathbb{E}\left[y_{i,a} \mid X\right] \\
&= \mathbb{E}\left[\frac{1}{N_F}\sum_{i=1}^{N_F}\sum_{a\in\mathcal{A}_i}\mathbb{1}(\tau^*(i)=a)y_{i,a} \mid X\right],
\end{aligned}
\tag{A.20}
$$

where the last term is the true match value under the efficient targeting policy. In the first two lines in Equation (A.20), we just use the linearity of the expectation function. In the third line, we use the unconfoundedness assumption. In the last two lines, we again use the linearity of the expectation function.

Now, we use the specification in Equation (A.20) to estimate the match value under the efficient targeting policy. We find that the efficient targeting policy coincides with the ads shown in the Filtered Sample in 67,152 impressions. We use inverse propensity weights to adjust for this fraction of the sample and find that the efficient targeting policy improves the average CTR in the current regime by 65.53%. This is very similar to our original finding using a direct method based on our match value estimates.

E Appendix for Robustness Checks

E.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 intuitive metric to evaluate the performance of our model.

We now take the optimized models presented in §5 in the main text of the paper 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.

E.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.

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.

- 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.
- 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

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.

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: $\{n_{tree} = 1000, \chi_s = 0.33, n_{\min} = 500\}$.

E.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*.

E.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.[5] 3) We then calculate the mean and standard deviation of the *RIG* for each sample. We 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

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

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.

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.

E.5 Other Validation Techniques

The validation procedure outlined in Web 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 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, 2020) and (Hastie et al., 2001) for a detailed discussion on the pros and cons of k-fold cross validation.

F Detailed Analysis of the Example Presented in Figure 8 in the Main Text

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

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 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: $S = 0.6$	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: $R = 0.1 + 0.1 = 0.2$ 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.

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.21). 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 $u^{(2)}$.

$$eCTR = \begin{array}{c} \\ a^{(1)} \\ a^{(2)} \end{array} \begin{array}{c} \begin{array}{cc} u^{(1)} & u^{(2)} \end{array} \\ \begin{pmatrix} 0.5 & 0.1 \\ 0.1 & 0.3 \end{pmatrix} \end{array} \longrightarrow \begin{array}{c} \bar{u} \\ \begin{pmatrix} 0.3 \\ 0.2 \end{pmatrix} \end{array} \tag{A.21}$$

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 H 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 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.

G Proofs

G.1 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 regime, 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 in the main text of the paper, for any $I_k^{(2)} \in \mathcal{I}^{(2)}$, there exist $I_{j_1}^{(1)}, \ldots, I_{j_l}^{(1)}$ such that $\bigcup_{s=1}^{l} I_{j_s}^{(1)} = I_k^{(2)}$. We can write:

$$\max_a \sum_{i \in I_k^{(2)}} v_{ia} \leq \sum_{s=1}^{l} \max_a \sum_{i \in I_{j_s}^{(1)}} v_{ia} \tag{A.22}$$

where the LHS is the surplus generated from the impressions in bundle $I_k^{(2)}$ for targeting regime

$\mathcal{I}^{(2)}$, and RHS is the surplus generated from the same impressions for targeting regime $\mathcal{I}^{(1)}$. Since inequality (A.22) 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.22) 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.2 Proof of Proposition 2

To write the FOC, we first need to specify advertisers' expected utility function. If ad a gets a click in impression i, the utility he extracts from this impression is $v_a^{(c)} - \textit{CPC}_{ia}$. As such, ad a's utility from impression i is

$$\left(v_a^{(c)} - \textit{CPC}_{ia}\right) \mathbb{1}(a_i = a)\mathbb{1}(y_{i,a} = 1),$$

where $\mathbb{1}(a_i = a)$ indicates whether ad a is shown in impression i and $\mathbb{1}(y_{i,a} = 1)$ is an indicator for whether this impression is clicked. Clearly, ad a only extracts utility when his ad is being shown and clicked in an impression. We further assume that $\mathbb{1}(a_i = a)$ is independent of CPC_{ia} with respect $\mathcal{G}_a$. Now, we define the expected utility function $\textit{EU}_a$ as the expectation over impressions as follows:

$$\begin{aligned} \textit{EU}_a(b_a; \mathcal{G}_a) &= \mathbb{E}_{\mathcal{G}_a}\left[\left(v_a^{(c)} - \textit{CPC}_{ia}\right) \mathbb{1}(a_i = a)\mathbb{1}(y_i = 1)\right] \\ &= \left(v_a^{(c)} - c_a(b_a)\right) \pi_a(b_a)\mu_a \\ &= \left(v_a^{(c)} - c_a(b_a)\right) \frac{b_a q_a}{b_a q_a + Q_{-a}}\mu_a \end{aligned} \tag{A.23}$$

where $\mu_a = \mathbb{E}_{\mathcal{G}_a}\left[\mathbb{1}(y_i = 1)\right]$ is the average probability of click that ad a expects to receive. The second line of Equation (A.23) is resulted because of the independence of $\mathbb{1}(a_i = a)$ and CPC_{ia} with respect $\mathcal{G}_a$.

Now, to write the FOC, we need to take the first derivative of the expected utility function for ad

a. We can write:

$$\frac{\partial EU_a(b_a;\mathcal{G}_a)}{\partial b_a} = \left(v_a^{(c)}\pi_a'(b_a) - c_a(b_a)\pi_a'(b_a) - c_a'(b_a)\pi_a(b_a)\right)\mu_a \tag{A.24}$$

Now, to satisfy the FOC, we need to have $\frac{\partial EU_a(b_a^*;\mathcal{G}_a)}{\partial b_a^*} = 0$, where b_a^* is the equilibrium bid. Using Equation (A.24), we can write the FOC as follows:

$$\begin{aligned} v_a^{(c)} &= c_a(b_a^*) + \frac{c_a'(b_a^*)\pi_a(b_a^*)}{\pi_a'(b_a^*)} \\ &= c_a(b_a^*) + \frac{c_a'(b_a^*)b_a^*}{1-\pi_a(b_a^*)}, \end{aligned} \tag{A.25}$$

where the second line is because we have $\frac{\pi_a(b_a)}{\pi_a'(b_a)} = \frac{b_a}{1-\pi_a(b_a)}$ which is easy to show:

$$\frac{\pi_a(b_a)}{\pi_a'(b_a)} = \frac{\frac{b_a q_a}{b_a q_a + Q_{-a}}}{\frac{q_a Q_{-a}}{(b_a q_a + Q_{-a})^2}} = \frac{b_a(b_a q_a + Q_{-a})}{Q_{-a}} = \frac{b_a}{1-\pi_a(b_a)} \tag{A.26}$$

Now, to complete the proof, we need to show that the second-order condition (SOC) is also satisfied. We start by writing a useful property in the relationship between the first and second derivative of the allocation function:

$$\frac{\pi_a'(b_a)}{\pi_a''(b_a)} = \frac{\frac{q_a Q_{-a}}{(b_a q_a + Q_{-a})^2}}{\frac{-2q_a^2 Q_{-a}}{(b_a q_a + Q_{-a})^3}} = \frac{(b_a q_a + Q_{-a})}{2q_a} = -\frac{b_a}{2\pi_a(b_a)} \tag{A.27}$$

We can now write the second derivative of the expected utility function as follows:

$$\begin{aligned}
\frac{\partial^2 EU_a(b_a^*;\mathcal{G}_a)}{\partial {b_a^*}^2} &= \mu_a \left(\left(v_a^{(c)} - c_a(b_a^*) \right) \pi_a''(b_a^*) - c_a'(b_a^*)\pi_a'(b_a^*) - c_a'(b_a^*)\pi_a'(b_a^*) - c_a''(b_a^*)\pi_a(b_a^*) \right) \\
&= \mu_a \left(\frac{c_a'(b_a^*)\pi_a(b_a^*)}{\pi_a'(b_a^*)} \pi_a''(b_a^*) - 2c_a'(b_a^*)\pi_a'(b_a^*) - c_a''(b_a^*)\pi_a(b_a^*) \right) \\
&= \mu_a \left(c_a'(b_a^*)\pi_a(b_a^*) \left(\frac{\pi_a''(b_a^*)}{\pi_a'(b_a^*)} - 2\frac{\pi_a'(b_a^*)}{\pi_a(b_a^*)} \right) - c_a''(b_a^*)\pi_a(b_a^*) \right) \\
&= \mu_a \left(c_a'(b_a^*)\pi_a(b_a^*) \left(\frac{2\pi_a(b_a^*)}{b_a^*} - 2\frac{1-\pi_a(b_a^*)}{b_a^*} \right) - c_a''(b_a^*)\pi_a(b_a^*) \right) \\
&= \mu_a \left(-\frac{2}{b_a^*} c_a'(b_a^*)\pi_a(b_a^*) - c_a''(b_a^*)\pi_a(b_a^*) \right) \\
&= \mu_a \pi_a(b_a^*) \left(-\frac{2}{b_a^*} c_a'(b_a^*) - c_a''(b_a^*) \right)
\end{aligned} \tag{A.28}$$

In the equation above, we know that both μ_a and $\pi_a(b_a^*)$ are positive, so it is sufficient to show that $-\frac{2}{b_a^*} c_a'(b_a^*) - c_a''(b_a^*) \leq 0$, which is resulted from the assumption in Proposition 2 in the main text of the paper. Thus, the SOC is satisfied and this completes the proof.

H 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 $\max_a v_{ia} m_{ia}$. Thus, defining b_{ia} as advertiser a's bid on impression i, the platform should sell the ad slot to $\operatorname{argmax}_a b_{ia} m_{ia}$ 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.)

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: $R = 0.66 \times (0.5 + 0.1) = 0.4$	Platform's expected revenue: $R = 0.2 \times 0.5 + 0.33 \times 0.3 = 0.2$
Advertiser's expected surplus: $W_1 = (1 - 0.66) \times (0.5 + 0.1) = 0.2$ $W_2 = 0$	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: $S = 0.6$	Total expected surplus: $S = 0.8$

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

Table A6 shows how the example in §6.1 in the main text of the paper 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 regime 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.

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

I.1 Alternative Methods for Estimation of Click Valuations

As discussed in §6.3.2 in the main text of the paper, we make three simplifications to derive Equation (15) in the main text of the paper.

1. **Simplification 1:** Advertisers' probability of winning an impression is approximately zero, i.e., $\pi_a(b_a) \approx 0$, for all a.

2. **Simplification 2:** Advertisers' cost-per-click is approximately their bid, i.e., $c_a(b_a) \approx b_a$, for all a.

3. **Simplification 3:** The first-order condition shown in Equation (14) in the main text of the paper is satisfied for all advertisers, including reserve price bidders.

If one or more of these simplifications fail in our data, our estimates of click valuations may be biased. This, in turn, can affect our main qualitative findings. Therefore, we now present six alternative methods to estimate click valuations that progressively relax these simplifications and show that our main findings are robust to these simplifications.

Table A7 gives an overview of the set of simplifications that each alternative method relaxes. We now describe each of these methods in detail.

1. **Alternative Method 1 (AM1):** In this method, we relax Simplification 1, but retain Simplifications 2 and 3.

   Specifically, instead of assuming that $\pi_a(b_a) \approx 0$, we estimate the advertiser's expected probability of ad allocation directly from the data as:

$$\hat{\pi}_a(b_a^*) = \frac{\sum_{i=1}^{N_F} \mathbb{1}(a_i = a)}{\sum_{i=1}^{N_F} e_{i,a}}. \tag{A.29}$$

Simplification	Main	AM1	AM2	AM3	AM4	AM5	AM6
Zero winning probability ($\pi_a \approx 0$)	✓	×	×	×	×	×	×
Pay-your-bid cost function ($c_a(b_a) \approx b_a$)	✓	✓	×	×	×	×	×
First-order condition satisfied for reserve price bidders	✓	✓	✓	✓	×	×	×
Parametric cost function	✓	✓	✓	✓	✓	✓	×

Table A7: Table of simplifications imposed by each estimation method

This is the total number of impressions showing ad a divided by the total number of impressions ad a bid on (i.e., participated in the auction for). This is a consistent estimator of the proportion of impressions that ad a will win in the platform.

Since we still assume that advertisers pay their own bid per click (Simplification 2), we have $c_a(b_a) = b_a$. This gives us the click value estimates as:

$$\hat{v}_a^{(AM1)} = b_a^* + \frac{b_a^*}{1 - \hat{\pi}_a(b_a^*)}, \tag{A.30}$$

2. **Alternative Method 2 (AM2):** In this method, we relax both Simplifications 1 and 2. Like AM1, here also we estimate the probability of winning (proportion of impressions allocated) from the data using Equation (A.29). In addition, we no longer assume that advertisers pay their bid. Instead, we model their payment upon winning as a linear function of their bid such that $c_a(b_a) = \gamma_a b_a$. Then, using Equation (14) from the main text of the paper, we can write:

$$\hat{v}_a^{(AM2)} = \hat{c}_a(b_a^*) + \frac{\hat{c}_a(b_a^*)}{1 - \hat{\pi}_a(b_a^*)} \tag{A.31}$$

This is because if $c_a(b_a)$ is linear, we have $c_a(b_a) = c_a'(b_a)b_a$. Hence, we only need to estimate $\hat{c}_a(b_a^*)$, which is the average cost-per-click for ad a.

3. **Alternative Method 3 (AM3):** This method is similar to AM2. The main difference is that, here we model the cost function more flexibly by allowing it to be a quadratic function of bids. We can characterize our cost function as $c_a(b_a) = \gamma_a b_a + \gamma b_a^2$. Note that the coefficient for b_a is ad specific, but the one for b_a^2 is the same for all ads. The is because we cannot identify advertiser-specific curvature since advertisers do not change their bids.

We can estimate this quadratic cost function from the data using the following regression model:

$$\textit{CPC}_i = \gamma_a b_a + \gamma b_a^2 + \epsilon_i, \tag{A.32}$$

where CPC_i is the actual cost-per-click for impression i.

Once we have the estimates, $\hat{\gamma}_a$ and $\hat{\gamma}$, we can estimate click valuations using Equation (14) from the main text of the paper as follows:

$$\hat{v}_a^{(AM3)} = \hat{c}_a(b_a^*) + \frac{b_a^*(\hat{\gamma}_a + 2\hat{\gamma}b_a^*)}{1-\hat{\pi}_a(b_a^*)} = \hat{c}_a(b_a^*) + \frac{\hat{c}_a(b_a^*)}{1-\hat{\pi}_a(b_a^*)} + \frac{\hat{\gamma}b_a^*}{1-\hat{\pi}_a(b_a^*)} \tag{A.33}$$

Equation (A.33) is very similar to Equation (A.31), except for the last term $\frac{\hat{\gamma}b_a^*}{1-\hat{\pi}_a(b_a^*)}$, which is added to incorporate the curvature in the relationship between bid and cost-per-click. Of course, if the relationship is linear, (A.33) will be identical Equation (A.31).

While both AM2 and AM3 allow the cost-per-click to differ from the actual bid, they formulate an advertiser's cost-per-click only as a function of their own bids, and not that of its competitors. Given Proposition 2, our click valuation estimates are valid to the extent that our parametric cost functions estimate $c_a'(b_a^*)$ well. Nevertheless, in AM6, we estimate the advertisers' cost function without imposing any functional form assumptions, and allow it to depend on other advertisers' bids as inputs.

4. **Alternative Method 4 (AM4):** In this method, we relax all three simplifications. We build on AM3 and incorporate the fact that there is a reserve price r_0 for all impressions (i.e., advertisers need to bid higher than or equal to this amount).

   In the presence of a reserve price, equilibrium bids may not satisfy the first-order condition since they may be right-censored. That is, there could have been bids lower than r_0 without the reserve price. For reserve-price 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 $v_a^{(c)} \leq r_0\left(1 + \frac{1}{1-\pi_a(b_a)}\right)$. Thus, for reserve bidders, we have $v_a^c \in [r_0, r_0(1 + \frac{1}{1-\pi_a(b_a)})]$. As such, if such valuations come from a uniform distribution in this range, our estimates will be:

$$\hat{v}_a^{(AM4)} = \begin{cases} r_0\left(1 + \frac{1}{2(1-\hat{\pi}_a(b_a^*)}\right) & \text{if } a \text{ is a reserve bidder} \\ \hat{v}_a^{(AM3)} & \text{otherwise} \end{cases}, \tag{A.34}$$

   where the estimated click valuations for a reserve bidder a is the mean of uniform distribution $\mathcal{U}\left(r_0, r_0\left(1 + \frac{1}{1-\hat{\pi}_a(b_a^*)}\right)\right)$.

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

such, our estimates will be:

$$\hat{v}_a^{(AM5)} = \begin{cases} x \sim \mathcal{U}(r_0, r_0(1 + \frac{1}{1-\hat{\pi}_a(b_a^*)})) & \text{if } a \text{ is a reserve bidder} \\ \hat{v}_a^{(AM3)} & \text{otherwise} \end{cases} \tag{A.35}$$

6. **Alternative Method 6 (AM6):** Here, we implement the the non-parametric estimator proposed by Rafieian (2020) for quasi-proportional auctions. This is a structural approach to estimate click valuations that relaxes all the simplifications made earlier and incorporates the full structure of the setting. The main difference between his approach and the alternative methods AM1–AM5 is in the operationalization of the cost function. Unlike the earlier methods, which parameterized the cost function, he employs a fully non-parametric method for modeling the cost distribution. As such, advertisers' expected cost function is obtained by taking the expectation over the distribution of auctions they participate in. We refer readers to Rafieian (2020) for more details.

Results from Alternative Methods for Click Value Estimation

We first illustrate the empirical CDFs of all these alternative methods as well as the main method used in the paper. As we can see from Figure A.5, the estimated value distributions are not very different. This is mostly due to the fact that Simplification 1 is mostly valid (i.e., there are many bidders and no one has disproportionately high chance of winning), which in turn, makes the approximation method used in Equation (15) in the main text quite reasonable.

Next, we use the click value estimates from each of these alternative methods to estimate the market outcomes – total surplus, platform revenues, and advertisers' surplus. The results are shown in Table A8. While these estimates are quantitatively different from those presented in Table 5 in the main text of the paper, notice that our main qualitative results remain the same: we find a monotonic relationship between total surplus and the granularity of targeting, but platform revenues are maximized when the platform limits the targeting level to the contextual targeting. In particular, we expect that results from AM6 to be the most accurate since it is the most assumption-free method, and even here the main qualitative finding holds.

Further, we find that the results from AM1 are the closest to the results in the main text (Table 5 in the main text of the paper). This is because AM1 still uses the approximation that $CPC_a \approx b_a$. On the other hand, the results from AM2 to AM6 are very close quantitatively. This is because they all relax Simplification 2 and estimate the cost function from the data. As a result, the magnitude of total surplus and platform revenues is smaller (because advertisers in a quasi-proportional auction submit higher bids when they know they are not exactly charged their submitted bids). However,

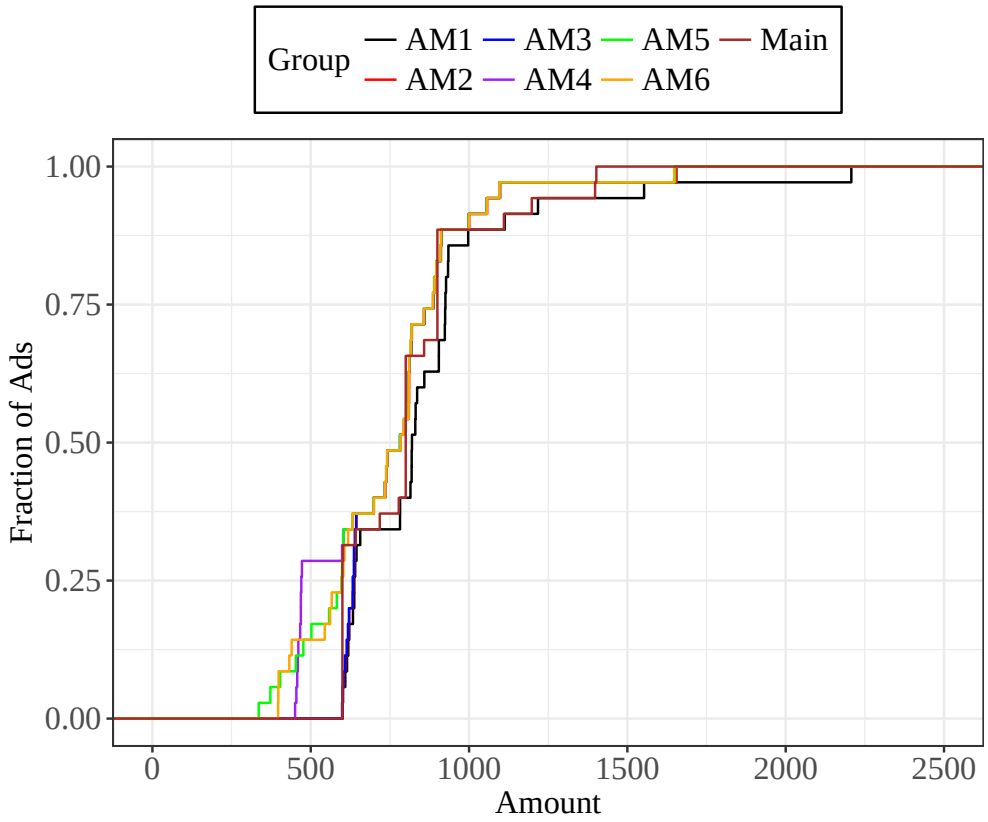

Figure A.5: Empirical CDF of estimated value distributions using alternative methods

the main qualitative findings remains the same under all these methods, since the main source for these findings is the heterogeneity in match valuations induced by more granular targeting.

I.2 Adding Noise to Match Valuations

As discussed in §6.3.3 in the main text of the paper, 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.

I.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:

$$\hat{m}_{i,a}^{(\nu)} = \hat{m}_{i,a}\epsilon_{i,a}, \tag{A.36}$$

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	8.81	8.04	0.77
Full Targeting	9.08	8.04	1.04
E. 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	8.04	1.04
D. 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	8.03	1.05
F. Alternative Method 6 (AM6)			
No Targeting	8.03	7.91	0.12
Contextual Targeting	8.61	8.08	0.53
Behavioral Targeting	8.82	8.03	0.78
Full Targeting	9.08	8.04	1.04

Table A8: 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.

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 §6.3.4 in the main text of the paper to estimate the average surplus, revenue, and advertisers' surplus. The results are shown in Table A9. We find that noise-adding can increase the 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

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 A9: 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.

same results when $\nu = 0.3$. Thus, we can interpret contextual targeting as a noise-adding strategy.

I.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:

$$\hat{m}_{i,a}^{(\nu_a)} = \hat{m}_{i,a}\epsilon_{i,a}, \tag{A.37}$$

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 A10. Again, we find that the platform benefits when advertisers have imperfect estimates of their match valuations. Comparing the results in Table A9 and A10, 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 A10: 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.

D. F. McCaffrey, B. A. Griffin, D. Almirall, M. E. Slaughter, R. Ramchand, and L. F. Burgette. A tutorial on propensity score estimation for multiple treatments using generalized boosted models. *Statistics in medicine*, 32(19):3388–3414, 2013.

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.

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.

O. Rafieian. Revenue-optimal dynamic auctions for adaptive ad sequencing. 2020.

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

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. *Management Science*, 66(3):1045–1070, 2020.

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