Targeting and Privacy in Mobile Advertising

Omid Rafieian[*]
University of Washington

Hema Yoganarasimhan[*]
University of Washington

March 9, 2018

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

Abstract

Mobile in-app advertising is growing in popularity. While these ads have excellent user-tracking properties through mobile device IDs, they have raised concerns among privacy advocates. There is an ongoing debate on the value of different types of mobile targeting, the incentives of ad-networks to engage in behavioral targeting and share user-data with advertisers, and the role of regulation. To answer these questions, we propose a modeling framework that consists of two components – a machine learning framework for predicting click-through rate and a stylized analytical framework for conducting data-sharing counterfactuals. Using data from the leading in-app ad-network of an Asian country, we show that our machine learning model improves targeting ability by 17.95% over no targeting. These gains mainly stem from behavioral information and the value of contextual information is relatively small. Stricter regulations on user-tracking substantially shrink the value of behavioral targeting. Counterfactuals show that the total advertisers' surplus grows with more granular information-sharing between the ad-network and advertisers. However, there is heterogeneity among advertisers in their preferred level of data-sharing. Importantly, the ad-network's revenues are non-monotonic, i.e., it prefers to not share behavioral information with advertisers. Thus, the ad-network may have natural incentives to preserve users' privacy without external regulation.

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

1 Introduction

1.1 Mobile Advertising and Targeting

Smartphone adoption has grown exponentially in the last few years, with more than two billion people owning a smartphone today (eMarketer, 2017a). The average smartphone user now spends over 2.8 hours per day on her phone. In addition to making phone calls and browsing, users spend a significant amount of time on native applications, popularly known as "apps" (Perez, 2015; Chaffey, 2017). As mobile usage has surged, advertisers have followed the eyeballs. In 2016, mobile ad spend overtook desktop ad spend for the first time by generating over $106 billion dollars in revenues worldwide, which accounts for 55.7% of digital advertising revenues (eMarketer, 2017b).

The rapid growth of mobile advertising partly stems from an ad format unique to the mobile environment – "in-app ads" or ads shown inside apps. In-app advertising now generates over $72 billion dollars worldwide and owes its increasing popularity to two factors (AppAnnie, 2017). First, with the proliferation of free apps, in-app advertising has become a mainstream app monetization strategy (Hollander, 2017). Second, in-app ads have excellent tracking and targeting properties, allowing advertisers to "behaviorally" target their ads. Advertisers and ad-networks have access to a unique device ID associated with the mobile device that they can use for tracking and targeting purposes, referred to as IDFA (ID For Advertisers) in iOS devices and AAID (Android Advertiser ID) in Android devices. This device ID is highly persistent and remains the same unless actively re-set by the user. This allows advertisers to stitch together user data across sessions, apps, and even other advertisers (Edwards, 2012).[1]

While the advertising industry has lauded the trackability of in-app ads suggesting that they create value for advertisers by allowing them to target the right consumer, consumers and privacy advocates have derided them citing privacy concerns. Bowing to pressure from consumers, mobile phone makers have started taking steps to limit user-tracking. First, in 2012, Apple went from UDID (Unique Device ID) to IDFA. The latter is user re-settable (much like a cookie), whereas the former was a fully persistent unique device ID that could not be erased by users under any circumstance (Stampler, 2012). In 2013, Android made a similar move from Android ID to AAID (Sterling, 2013). While this caused some consternation among advertisers, it nevertheless did not upend the mobile advertising ecosystem since few users actively re-set IDFAs on a regular basis. However, in mid-2016, Apple took a much stronger stance in support of privacy by allowing users to completely opt out of tracking through LAT (Limit Ad Tracking), wherein a user can simply

[1]This is in contrast to browser-based mobile ads, which have poor tracking properties since third-party cookies don't work in many browsers (e.g., Safari, the most popular browser) and hence tend to be poorly targeted (Sands, 2015).

switch off tracking indefinitely (Gray, 2016).[2] This makes it impossible for advertisers to target ads, attribute conversions, or re-target and re-engage users. Studies show that nearly 20% of iOS users have already chosen to opt into LAT, raising significant concerns among advertisers about the future of targeted mobile advertising (Seufert, 2016).

This tension between behavioral targeting and privacy in mobile advertising is part of a larger ongoing debate over consumer tracking and privacy in digital contexts. Advertisers argue that tracking allows consumers to enjoy free apps and content, and see relevant ads, whereas regulators and consumers demand higher privacy, control over their data, and limits on data-sharing among firms. In mid-2016, the European Union signed the General Data Protection Regulation (GDPR) agreement, which will go into effect in May 2018 (Kint, 2017). Under the new regulations, consumers will have to opt into (rather than opt out of) behavioral targeting and will need to give explicit permission for their data to be shared across firms. Violators will be fined up to 4% of their global revenues. In the US, under the Obama administration, the Federal Communications Commission (FCC) adopted legislation to limit tracking and data-sharing among large ISPs. The advertising industry lobbied against these rules, and scored a win when these rules were recently relaxed under the Trump administration (Fung, 2017). Nevertheless, polls show that over 92% of US consumers believe that the government should ban tracking and sharing of consumer data (Edwards-Levy and Liebelson, 2017).

Even as this debate rages on, we do not have a good understanding of how tracking and data-sharing affect advertisers' ability to target, the incremental value of different types of information on improving targeting accuracy, and the incentives of different players in the mobile advertising industry. Lack of clear empirical analyses of these issues hampers our ability to have an informed discussion and to form policy on these issues.

1.2 Research Agenda and Challenges

In this paper, we are interested in three key research questions related to targeting and privacy in the mobile ecosystem.

- First, what is the value of different pieces of information in improving targeting accuracy? We are specifically interested in the relative value of contextual vs. behavioral information. The former quantifies the context (when and where) of an impression, and the latter constitutes historic data on an individual user's past app usage, ad exposure, and ad response. Contextual information is privacy preserving whereas behavioral information is based on user-tracking and therefore impinges on user-privacy. Further, we want to quantify the total improvement in

[2]When a user turns on LAT, her IDFA simply shows up as a string of zeroes to the advertisers. Thus, all the consumers who have opted into LAT are visible as one large ghost segment to ad networks.

targeting effectiveness that can be achieved when all types of information are used.

- Second, how would stronger privacy regulations affect the ad-network's ability to target? If the ad-network were to lose the ability to do user-tracking through device IDs (e.g., through blanket adoption of LAT or government regulation), to what extent would its targeting ability suffer?
- Finally, we are interested in examining the relationship between the advertisers' targeting ability and platform revenues. Does the ad-network have an incentive to share targeting data with advertisers and thereby enable targeted bidding? If yes, is there an optimal level of data-sharing?

To build a unified and scalable framework to answer these questions, we need to overcome three key challenges: 1) achieve high predictive accuracy, 2) have sufficient randomness in the data generating process, and 3) tie prediction to incentives. We discuss each of these in detail below.

First, to accurately quantify the value of different pieces of targeting information, we need to build a click prediction model with the best possible out-of-sample accuracy. That is, the model should be able to accurately predict whether a targeted impression will lead to a click or not. Indeed, models with poor predictive ability will lead to downward bias and noise in the estimates of the value of information. In the marketing literature, the commonly used approach to predictive modeling involves making functional form on the relationship between the outcome variable and the independent variables. From a statistical perspective, this can be thought of as parameter inference given a functional form. Implicitly, this allows the researcher to constrain the model space as well as the extent of interactions between variables. While this simplifies the problem that we need to solve, it also compromises the model's predictive ability because the researcher cannot intuit the appropriate functional-form a priori and indeed the correct functional form is likely to be extremely complex. Instead, the correct approach to building models with high predictive accuracy can be formally thought of as *function evaluation*. This allows for full flexibility in the interactions between variables as well as function-specification. However, this is also a much harder statistical learning problem to solve, especially when we have a large number of features.[3] Indeed, this problem is NP-hard and we need to turn to machine learning methods built for function evaluation to solve it.

Second, we need sufficient randomness in the data generating process for our predictive model to be applicable to counterfactual scenarios. Consider a hypothetical scenario where an ad is targeted to only women in the observed data. A well-specified machine learning model can generate accurate predictions of womens' CTR for this ad. However, it cannot tell us how men will respond to this

[3]We have around 160 features. So from a function evaluation perspective, we have to explore all the non-linear combinations of these features. This problem is exacerbated by the fact that we don't know which of these variables and which interaction effects matter, a priori. Even if we try to constrain the model to pairwise interactions, we would be working with 12960 variables. Thus, it is not feasible to come up with the best non-linear specification of the functional form either by manually eyeballing the data or by including all possible interactions.

ad since we never observe mens' response to it. Thus, if we want to use our predictive models to generate counterfactual predictions of CTRs for alternative targeting strategies, we need the model to have been trained on data that is not too narrowly targeted. Third, we need an underlying model of strategic interactions that can quantify effect of different targeting strategies on ad-network and advertiser revenues. Without an economic model of market players which puts some structure on revenues, we cannot make any statements on how ad-networks or advertisers' incentives to target and/or the extent to which these incentives are aligned.

Thus, we need a unified modeling framework that coherently combines predictive machine learning models with prescriptive economic models in order to overcome these challenges and answer our research questions of interest.

1.3 Our Approach

We present a modeling framework consists of two main components – (1) A machine learning framework for Click-Through Rate (CTR henceforth) prediction, and (2) a stylized analytical model to characterize the ad-network and advertisers' profits under different data-sharing (and corresponding targeting) scenarios.

- The machine learning part consists of two modules – (a) a feature generation framework and (b) a learning algorithm. The former relies on a set of functions to generate a large number of features that capture the contextual and behavioral information associated with an impression. Our feature functions can be aggregated over different lengths of history preceding the impression, thereby allowing us to capture the impact of long-term, short-term, and ongoing session-level history. Using these functions, we generate a total of 161 features. These serve as input variables into a CTR prediction model, which forms the second module. We use the Extreme Gradient Boosting algorithm (XGBoost) proposed by Chen and Guestrin (2016), a fast and scalable version of boosted regression trees, to train and infer the optimal the CTR prediction model. Our contribution here primarily lies in our feature generation framework that generates informative and theory-driven feature functions for the mobile in-app advertising ecosystem.
- The second component of our model is a framework for conducting data-sharing counterfactuals. We develop a stylized analytical model of data-sharing that characterizes the ad-network's revenue and advertisers' profit under different-levels of data-sharing. We analytically prove that while the total surplus in the system increases with more granular data-sharing and targeted bidding, the ad-network's revenue can go in either direction when we do not impose any functional form assumptions on the distribution of match values. We then combine this analytical model with the machine-learning targeting models developed in the first module to derive empirical estimates of the ad-network's revenue and advertisers' surplus under four

counterfactual data-sharing scenarios.

We apply our framework to one month of data from the leading mobile advertising ad-network from a large Asian country. The scale and scope of the data are large enough to provide realistic substantive and counterfactual results. For our analysis, we sample over 27 million impressions for training, validation, and testing, and use another 146 million impressions for feature generation.

A notable feature of our data is the extent of *randomization* of ads over impressions. Unlike other advertising ecosystems, in our setting ads are shown over a large variety of impressions and impressions can be allocated a wide variety of ads, i.e., the targeting/match between ads and impressions is quite low. Three unique structural aspects of the marketplace contribute to this feature. First, the ad-network uses a quasi-proportional auction mechanism, where the allocation rule is probabilistic rather than deterministic (Mirrokni et al., 2010). Thus, the highest scoring bidder does not always win. Second, the ad-network supports limited targeting and the extent of targeting in the ad-network is low. Third, the ad-network uses the average past CTR of an advertiser in the ad-network to generate the advertiser's score (bid $\times$ CTR). So the score is a weak predictor of the likelihood of a click. Together these lead to significant randomness in the allocation of ads to impressions. In principle, we do not need any randomness in the data generating process to build a good predictive model of clicks. All the substantive results from the machine learning module of our model are valid as long as the joint distribution of features and clicks is the same in the training and test data-sets. However, when we move to counterfactuals, we need our machine learning to be able to predict the CTR of not just the ad actually shown in an impression, but of all ads competing for that impression. In this case, having seen a large variety of ad-impression matches greatly enables our analysis. Thus, the second set of counterfactual results would not be possible if ads were not sufficiently randomized over impressions in the observed data.

1.4 Findings and Contribution

We now describe our main findings. Our results can be categorized into two groups – a first set of substantive results from the machine learning component of the model, and a second set of results from the what-if analysis that combines the analytical model of data-sharing with the machine learning targeting model.

We start with the substantive results from the machine learning model. First, we show that our full targeting model (that uses all the 161 features) can significantly improve advertisers' targeting ability. Our model leads to a 17.95% improvement in targeting ability compared to the base case of no targeting, as measured by Relative Information Gain (*RIG*). We also show that XGBoost outperforms other commonly used methods such as least squares, logistic regression, LASSO, single regression trees, and random forests. More broadly, models that employ data partitioning (e.g., tree-based algorithms) perform better than other model classes.

Second, we quantify how different pieces of information contribute to the efficacy of our targeting model. We first consider the relative value of contextual and behavioral information. We find that pure behavioral targeting (without using any contextual information) leads to a 12.14% improvement in *RIG* compared to the baseline model, whereas pure contextual targeting (without using any behavioral information) provides only 5.25% improvement in *RIG*. Taken together, our results can help managers gain intuition on when and where consumers click, which consumers click, and what types of ads they click on and estimate the benefits of targeting based on different features.

Third, we examine the impact of strengthening privacy regulations on the ad-network's ability to target. We consider a scenario where user-tracking through device IDs is banned (e.g., through blanket implementation of LAT) and the ad-network has to rely on the second-best user-identifier, IP address. In this case, the performance of our full targeting model drops to 13.38%. Moreover, we find that the relative value of contextual vs. behavioral information flips, i.e., contextual features contribute more to the model's performance than behavioral features based on IP. This suggests that behavioral targeting without a reliable user-identifier is not very effective. Next, we examine if the loss in behavioral information can be mitigated if we have longer user histories. Surprisingly, the efficacy of IP-based behavioral targeting worsens with the length of history. Thus, using longer histories can actually be counter-productive with weak identifiers.

We further investigate why IP is a weak identifier. Note that both IP and device IDs suffer from the problem of potentially identifying one user as multiple users because the user-identifier can be re-set. In the case of IP, this problem is more severe because it can be automatically re-set whereas device-IDs like AAID have to be actively re-set by the user. While this is one source of the problem, IP has a more serious flaw – that of incorrectly identifying multiple users as one user because they share the same IP. Our findings on the length of user history suggest that the problems with using IP as the user-identifier likely stem from the latter issue.

Finally, we address the question of ad-network's incentives –"Does the ad-network have incentive to share targeting data with advertisers and thereby enable targeted bidding? If yes, is there an optimal level of data-sharing? In an influential paper, Levin and Milgrom (2010) conjecture that high levels of targeting can soften competition between advertisers and thereby reduce ad-network's revenues. We flesh out their ideas in a stylized analytical model that uses second price auctions and characterize the ad-network's revenue, advertiser's surplus, and total surplus under different data-sharing scenarios. Our analytical model highlights the ad-network's trade-off between value creation and value appropriation when deciding the optimal level of data-sharing.

We take this model to data and consider four counterfactual data-sharing scenarios. We show that as the ad-network shares more granular targeting data, the total surplus increases (implying that targeting creates value by matching the best advertiser and impression). However, ad-network revenues are not monotonic; it does not increase with more granular data-sharing. Interestingly, we find that ad-network revenues are maximized when the ad-network restricts data-sharing to the contextual level, i.e., sharing behavioral information thins out the market, which in turn reduces ad-network revenues. Thus, ad-network may be incentivized to adopt a privacy-preserving data-sharing regime, especially if it cannot extract any additional surplus from advertisers through explicit data-sharing contracts. Next, we examine advertisers' surpluses. Although a majority of advertisers prefer a regime where the ad-network shares behavioral information, there is heterogeneity in their preferences. That is, some advertisers prefer more privacy-preserving data-sharing scenarios. Overall, given the structure of incentives of the ad-network and advertisers, it may not be necessary for an external entity such as EU/FCC to impose privacy regulations. To some extent, self-regulation by the industry is feasible.

In sum, our paper makes three main contributions. First, from a substantive perspective, we quantify the efficacy of targeting, present a comprehensive comparison of contextual and behavioral targeting, and the impact of stricter privacy regulations on advertisers' ability to target. Importantly, we identify the misalignment of the ad-network's and advertisers' incentives regarding behavioral and contextual information disclosure. Second, from a methodological perspective, we present a framework for building effective targeting models and conducting data-sharing counterfactuals. A key contribution of our paper lies in its combination of machine learning techniques with economic models. An important point we make is that while machine learning models are useful for pure prediction tasks (e.g., targeting), if we want to answer broader questions on economic importance (e.g., revenues under counterfactual strategies), we need some additional data attributes (at least some randomization in the data generating process) and a theory-driven economic models of strategic interactions between market players. Finally, from a policy perspective, we establish the preferences of the ad-network and advertisers on the optimal level of data-sharing. We expect our model and findings to be of interest to policy-makers interested in regulating data-sharing and user-tracking in the mobile advertising marketplace.

2 Related literature

First, our paper relates to the computer science literature on the engineering side of targeting – how to do targeting and how we can build models to predict click-through rates (CTR). A body of earlier works in that literature has developed probabilistic models of ad placement for both contextual and behavioral targeting (Broder et al., 2007; Chakrabarti et al., 2008; Chen et al., 2009; Ahmed et al., 2011). Another stream of computer science literature on targeting pertains to models of CTR prediction in online advertising and makes some prescriptive suggestions on feature generation, model selection, learning rates, and scalability (McMahan et al., 2013; He et al., 2014; Chapelle et al., 2015). Although we build our machine learning framework following their approaches, our work differs in two main ways. First, our goal is substantive – we seek to understand and quantify the impact of different types of information in mobile ad targeting, whereas the previous papers are mainly concerned with presenting methods for predicting clicks in a scalable fashion. Second, unlike these previous papers, we then use our model to conduct counterfactuals addressing important policy questions: what are the ad-network's incentives to share data with advertisers, and the implications of such sharing on revenues and consumer privacy.

The marketing and economics literature on targeting has focused on two issues. First, a series of papers focus on measuring the returns to targeting. Farahat and Bailey (2012) use a difference-in-difference estimator to decompose the impact of targeting into selection bias and treatment effects components. The second stream of work focuses on the interplay between privacy and targeting. Using data from a series of regime changes in advertising regulations, Goldfarb and Tucker (2011b) find that lowering targeting reduces consumer response rates. Goldfarb and Tucker (2011a) and Tucker (2014) highlight the perils of excessive targeting as consumers perceive increased targeting as a threat to their privacy. Please see Goldfarb (2014) for an excellent review of targeting in online advertising and Acquisti et al. (2016) for a detailed discussion of consumer privacy issues.

Next, our paper relates the growing body of literature on ad-network's incentives to target in marketing and economics. Early analytical papers in this area show that imperfect targeting can benefit firms by softening competition (Chen et al., 2001; Iyer et al., 2005). Levin and Milgrom (2010) were one of the first to conjecture the trade-off between value creation and market thickness whereby too much targeting can thin markets which in turn would soften competition and make the ad-network worse off. This is particularly the case when targeting information reveals that an arriving impression is the high value match for only one advertiser (Celis et al., 2014). This assumption leads to a set of theoretical predictions that ad-network revenues exhibit a non-monotonic pattern with the extent of targeting (Bergemann and Bonatti, 2011; Fu et al., 2012; Amaldoss et al., 2015; Hummel and McAfee, 2016).

In spite of the increasing interest from the theoretical side, there has been little empirical work on ad-network's incentives to do targeting. Yao and Mela (2011) present a structural model to estimate advertisers' valuations and show that targeting benefits both advertisers and the ad-network. Similarly, Johnson (2013) finds that both advertisers and publishers are worse off when the ad-network introduces stricter privacy policies that reduce targeting. However, Athey and Nekipelov

(2010) present a case study of two keywords and show that "coarsening" CTR predictions can help a search advertising ad-network generate more revenue. In a more recent paper, Lu and Yang (2015) study the ad-network's decision on the targeting breadth, defined as the level of comprehensiveness of the user's interest profile, and find that there is a nonlinear relationship between the ad-network revenues and the targeting breadth allowed by the ad-network.

Our work also relates to the literature on information-sharing. Pancras and Sudhir (2007) were one of the first in marketing to examine the incentives of data-intermediaries. They find that a monopolist data-intermediary has an incentive to sell its services using non-exclusive arrangements with downstream retailers and use the maximum history available to target consumers. Related to auctions, a set of theory papers shows that full information disclosure may not always be optimal for the auctioneer (Ganuza, 2004; De Corniere and De Nijs, 2016; Marotta et al., 2017). In our setting, the ad-network can be interpreted as a data-intermediary that sells advertising slots through auction. We consider different data-sharing strategies and show that without explicit transfers between the ad-network and advertisers, the ad-network has an incentive to restrict data.

Our work thus adds to the growing literature on applications of machine learning in marketing (Toubia et al., 2003; Evgeniou et al., 2007; Dzyabura and Hauser, 2011; Huang and Luo, 2016; Liu and Dzyabura, 2016). Our paper closely relates to Yoganarasimhan (2017), who presents a case-study of personalized search and examines how heterogeneity in user-history and query-type can affect returns to personalization. We adopt many aspects of her approach such as the feature-generation functional framework, her data preparation techniques that takes advantage of user-level history, and boosted trees for training.

3 Setting and Data

3.1 Setting

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

3.1.1 Players

We now describe the four key players in this marketplace.

- Consumers: individuals who use apps. They see the ads shown within the apps that they use and may choose to click on the ads.
- Advertisers: firms that show ads through the ad-network. They design banner ads and specify their bid as the amount they are willing to pay per click, and can include a maximum budget if

they choose to. Advertisers can target their ads based on the following variables – app category, geographical location, connectivity type, time of the day, mobile operators, and mobile brand of the impression. The ad-network does not support more detailed targeting at this point in time.

- Publishers: app developers who have joined the ad network. They accrue revenues based on the clicks generated within their app. Publishers earn 70% of the cost of each click in their app (paid by the advertiser), and the remaining 30% is the ad-network's commission.
- Ad-network or Platform[4]: functions as the matchmaker between users, advertisers, and publishers. It runs a real-time auction for each impression generated by the participating apps and shows the winning ad during the impression. The platform uses a CPC pricing mechanism, and therefore generates revenues only when clicks occur. [5]

3.1.2 Auction Mechanism

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

$$\pi_a = \frac{b_a m_a}{\sum_{j \in \mathcal{A}^F} b_j m_j} \tag{1}$$

where π_a is the probability that advertiser a with bid b_a and quality score m_a wins the auction when the the set of advertisers is $\mathcal{A}^F$. Currently, the platform simply aggregates all total past impressions and clicks for an ad, and uses the ad-specific eCTR (expected Click Through Rate) as the quality score (m_a) in its auctions. After each impression, the ad-specific eCTR is updated based on whether the ad was clicked or not. Thus, the extent of customization in the quality scores is quite low. Further, the ad that can generate the highest expected revenue for the platform (the one with the highest $b_a m_a$) is not guaranteed to win. Rather, an ad's probability of winning is proportional to the expected revenue generated from it.

Quasi-proportional auctions have some advantages compared to the standard second price auction. While it is well-known that a second-price auction with optimal reserve prices is revenue optimal (Myerson, 1981; Riley and Samuelson, 1981), setting optimal reserve prices requires the auctioneer to know the distribution of valuations within each auction. This is not feasible when the valuations are changing constantly and/or the bidders in the system vary widely, as is commonly the case in online ad auctions. This is especially the case with our platform where the market is changing

[4]Throughout the paper, ad-network and platform are used synonymously.

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

significantly and advertisers are learning their valuations and responding to them as the marketplace evolves. In a *prior-free* setting such as this, Mirrokni et al. (2010) show that quasi-proportional auctions offer better worst-case performance than second-price auctions, especially when bidder valuations are starkly different.[6] Second, it ensures that individual consumers are not exposed to the same ad repeatedly. In contrast, in a deterministic auction, the same advertiser would win all the impressions unless his budget runs out. This can be irritating to consumers in mobile app settings because they would see the same ad for many consecutive impressions. For these reasons, the platform has adopted a quasi-proportional auction mechanism and does not employ a reserve price.

3.2 Data

We have data on all the impressions and corresponding clicks (if any) in the platform, by all the participating apps for a one month period from 30 September 2015 to 30 October 2015. Each impression in the data comes with the following information.

- Time and date: The time-stamp of the impression.
- IP address: The Internet Protocol address associated with the impression, which is essentially the IP of the accessing user's smartphone when the impression occurs.
- AAID: Android Advertising ID is a user re-settable, unique, device ID that is provided by the Android operating system.[7] It is accessible to advertisers and ad networks for tracking and targeting purposes. We use it as the user-identifier in our main analysis.
- App ID: A unique identifier for apps that advertise through the platform.
- Ad ID: This is an identifier for ads that are shown to the users.
- Bid: The bid that the advertiser has submitted for her ad.
- Location: The exact location of a user based on latitude and longitude.
- Connectivity type: It refers to the user's type of connectivity (e.g., WiFi or cellular data)
- Smartphone brand: The brand of user's smartphone (e.g., Samsung, Huawei, etc.)
- MSP: The user's mobile-phone service provider.

[6]Consider a setting with two bidders A and B, where A has a valuation of \$100 and B \$1. In this case, if the auctioneer has no prior knowledge of the distribution of valuations, he cannot set an appropriate reserve price. Without a reserve price, A will win the auction and pay \$1 in a second price auction, which is significantly lower than her valuation.

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

- ISP: The user's Internet service provider.
- Click indicator: This variable indicates whether or not the user has clicked on the ad.

The total data we see in this one month interval is quite large. Overall, we observe a total of 1,594,831,699 impressions and 14,373,293 clicks in this time-frame, implying a 0.90% CTR.

3.3 Data Preparation

3.3.1 Validation

Our goal is to develop a model that can accurately predict the CTR for a new impression. To achieve this goal, we specify and train a supervised machine learning algorithm. These learning algorithms typically require three steps (and corresponding data-sets) – training, validation, and testing (Hastie et al., 2001). Training is the process of constructing our prediction rule using data. However, as the prediction rule evolves to captures more complex relationships between variables, it is likely to over-fit on the training data. As a result, we might end up picking a model with great in-sample performance, but poor out-of-sample performance if we only use training data for model selection. Validation helps us avoid this problem by using parts of the data (referred to as the validation data) to validate the model and ensure that it will have a good out-of-sample performance. Since both training and validation data are used in model specification and selection, we use a completely new data, test data, to evaluate the model's out-of-sample performance . Please see Appendix §A for a detailed description of our validation and model selection procedure. Further, in §5.4 we show that our model selection and results are robust to the validation procedure used.

3.3.2 Data Splits

We have a snapshot of one month of data, from September 30 to October 30. We use the XX two days (October 28 and 29) for training and validation, and the last day for testing (October 30). We also use the preceding history from September 30 to Oct 27 (referred to as global data) to generate the features associated with these impressions. The splits of data are shown in Figure 1. Note that we do not fit our model on the global data because we do not have sufficient history to generate features for these impressions. Further, constraining all the three data-sets – training, validation, and testing – to a three-day window has advantages because recent research has shown that data freshness plays an important role in CTR prediction, i.e., using older history for prediction (and ignoring more recent history) can lead to poor predictive performance (He et al., 2014).

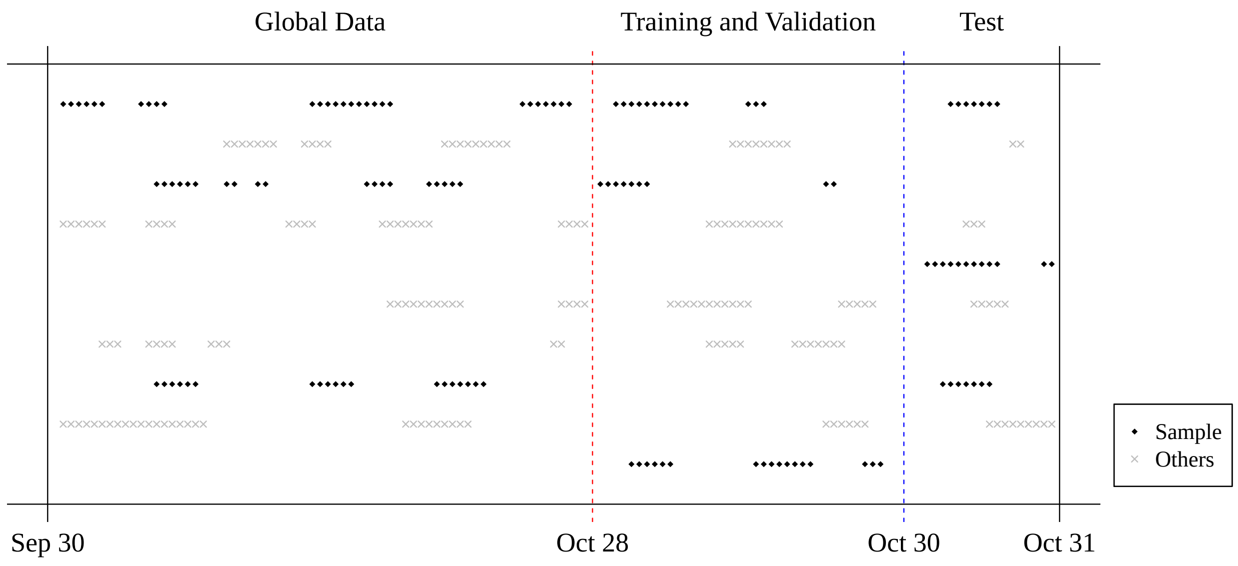


Figure 1: Schema for data generation.

3.3.3 Sampling

We draw a sample of 728,340 unique users (out of around 5 million) seen on October 28, 29, and 30 to form our training, validation, and test datasets.[8] We then track the impressions containing these users for the last one month to build the global data and generate the associated features for training, validation, and test datasets. In Appendix C.3, we show that this sample size is sufficient and that increasing the size of the sample further does not significantly improve model performance.

Figure 1 presents a visual depiction of the sampling procedure. Rows represent different users. We sample some users (rows in black point) and track them over the last one month and generate the global data. All the users in global data must be once present in either training and validation, or test data. In total, there are 146,825,916 impressions in global data, 17,856,610 impressions in the training and validation data, and 9,625,835 impressions in the test data.

Note that both our user-based sampling procedure and feature generation approach (see §4.2) require us to be able to identify and track users. For this purpose, we use the AAID variable as our user identifier. In §5.3, we further examine the value of this user identifier and discuss what would happen if we did not have access to this variable.

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

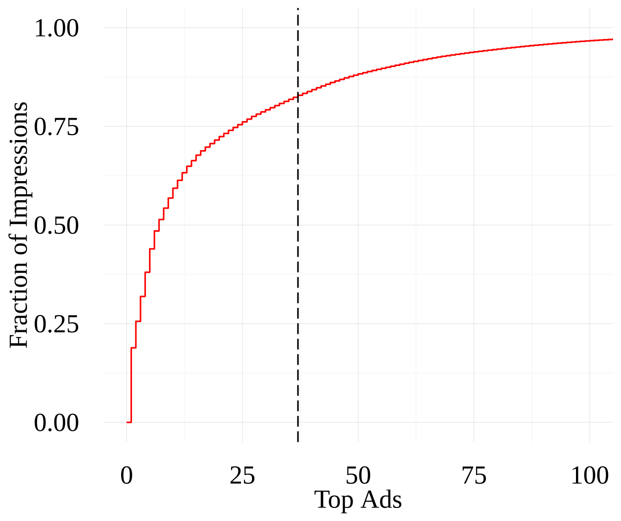
Variable	**Number of sub-categories**	**Share of top sub-categories** 1^{st}	2^{nd}	3^{rd}	**Number of impressions**
App	9709	37.12%	13.56%	3.05%	27,482,444
Ad	263	18.89%	6.71%	6.31%	27,482,444
Hour of the Day	24	7.39%	7.32%	6.90%	27,482,444
Smartphone Brand	8	46.94%	32.30%	9.53%	25,270,463
Connectivity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

Table 1: Summary statistics for the categorical variables.

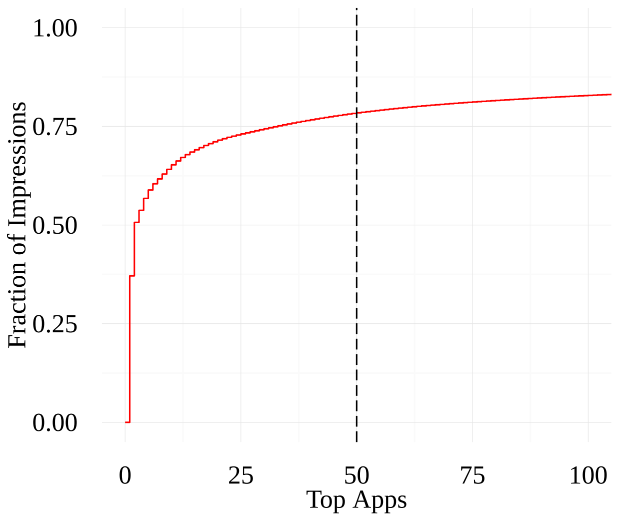
3.4 Summary Statistics

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

Table 1 shows the preliminary statistics on the categorical variables in the data. For each variable, we present the number of unique values, the share of top three sub-categories, and the number of non-missing data for each variable. While we always have information the app, ad, and time-stamp of the impression, the other variables are sometimes missing. The shares are shown after excluding the missing variables in the respective category.



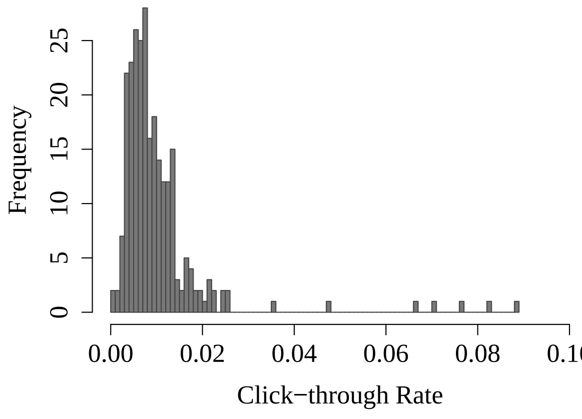
(a) Ads



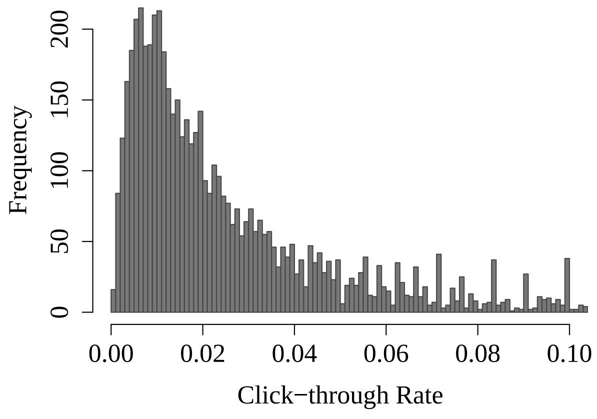
(b) Apps

Figure 2: Cumulative fraction of impressions associated with the top 100 ads and top 100 apps.

Next, we present some detailed statistics on ads, apps, and user-behavior. We observe a total of 263 unique ads and 9709 unique apps in the data. Further, the top three sub-categories in each have large shares and there is a long tail of smaller apps and ads. The top 37 ads account for over 80% of



(a) Ads



(b) Apps

Figure 3: Histogram of Click-through Rate for Ads and Apps

the impressions (see Figure 2a). Similarly, the top 50 apps account for 80% of impressions (see Figure 2b). Therefore, we focus on these top 37 ads and top 50 apps and group the smaller ads and apps into separate categories in our feature generation. It helps us reduce the dimensionality and boosts the speed of our feature generation framework.

Figure 3a and 3b show the histogram of CTRs for ads and apps respectively. The variance of CTRs among apps is higher than the variance among ads. An important factor driving this pattern is the lack of targeting in the platform. While users self-select into apps, ads are assigned to users more randomly due to the probabilistic nature of the auction. Therefore, we observe less variation in ads' CTR compared to that of apps. Later in §5.2.2, we show to what extent each piece of information helps us predict the click-through rates of impressions.

Next we examine the association between the length of user-history and CTR. Preliminary analysis suggests that users are becoming less sensitive to ads with time. Figure 4 shows the CDF of the length of history for all the impressions and clicks. Most of the clicks come from users with shorter history, while most impressions come from users with longer history. Users who have seen less than 500 impressions generate around 50% of all impressions, whereas they contribute to over 75% of the clicks. Thus, exploiting user-history information can help us explain the clicking behavior observed in data.

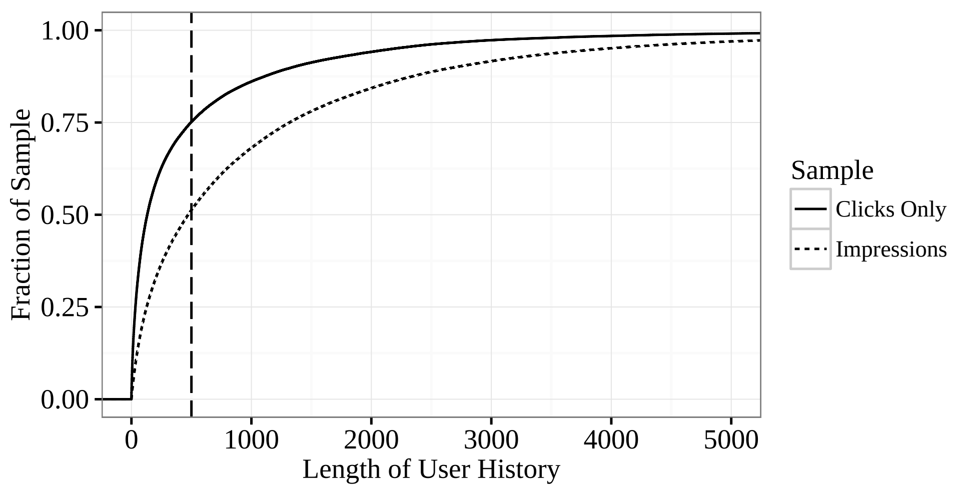


Figure 4: Empirical CDF of the length of user history for impressions and clicks (truncated at 5000). History is defined as the number of previous impressions from September 30 till the current impression (excluding it).

3.5 Randomness in the Data Generation Process

We now describe the importance and role of randomness in our model and present some empirical evidence on the extent of randomization observed in the data.

Randomization in data is not necessary to build a good predictive model of clicks. As long as the joint distribution of covariates and outcome (click) in the training set is the same as that in the test set, we can build a good predictive model. In other words, if we do not use the model to predict in counterfactual situations that are "not seen in the data", the lack of randomization is not a problem. Thus, even without randomization, all the substantive results in §5 are valid because there is no change in the system and the training and test sets have the same joint distribution of covariates and clicks.

However, in §6, we consider counterfactual targeting scenarios and predict the CTR of not just the ad actually shown in a given impression, but of all the ads competing for that impression. In such cases, our predictions of CTRs for the competing ads in the focal impression will be inaccurate if these ads were never shown in impressions similar to the focal impression. This is mainly because the joint distribution of covariates and clicks in the counterfactual scenario is not the same with that in the training set. For example, if an ad is only shown in App X in our data, then no predictive model (ML or otherwise) will be able to accurately predict the ad's click probability in App Y. Therefore, we need to see sufficient randomness in the allocation of ads across impressions to ensure that the joint distribution of covariates and clicks in our counterfactual scenarios is the same with

Targeting Area	**% of Top Ads Targeting**
App Category	32.4%
City	35.1%
Connectivity Type	5.4%
Time of the Day	32.4%
MSP	2.7%
Mobile Brand	2.7%
ISP	2.7%

Table 2: Targeting decisions of the top 37 ads.

that in the training set.[9] Later in §6.3, we propose a filtering approach which approximate the joint distribution of covariates and clicks using the auction mechanism and overcome this issue.

There are three main factors that contribute to the randomness in our data. First, it is the probabilistic nature of the quasi-proportional auctions. Unlike deterministic auctions where the highest scoring bidder always wins, in quasi-proportional auctions all bidders have a chance of winning. Therefore, ads end up being shown over a wide range of impressions. Second, the CTR used to generate the score (bid $\times$ CTR) is simply the average CTR for the ad in the platform. So the score is a weak predictor of the likelihood of receiving a click. Again, this leads to a scenario where ads are shown in a broad set of apps, users, and situations. Third, there are only few categories over which advertisers can target their ads and the extent of targeting happening in the system is low. Table 2 shows the list of variables over which advertisers can now target and the percentage of top advertisers that target on each of these variables. Interestingly, more than half of the top ads do not target in any given targeting area and even those that do target, do so minimally. Moreover, the platform does not do any behavioral or user-level targeting. Thus, the extent of randomization of ads over impressions is quite high. Figure 5 shows the empirical CDF of the number of competing ads (among the top 37) for each impression in the test data among top ads per impression. Almost all the impressions have at least 10 ads competing for it and the median impression has 20 competitors. This pattern, together with the probabilistic auction mechanism, ensures that a large variety of ads are shown across impressions.

[9]However, unlike a causal model, we do not need the allocation of ads across impressions to be completely orthogonal or perfectly random. As long as the extent of randomness is sufficiently large to ensure that the joint distribution of counterfactual scenario is the same as that of training set, we can get reasonable CTR predictions for all the top ads for all impressions.

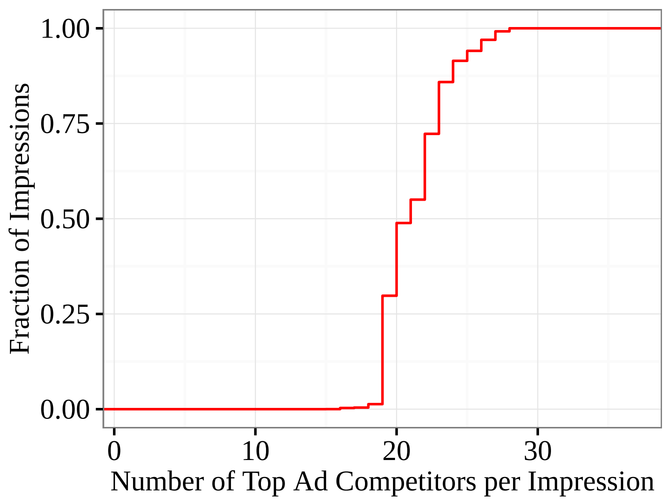


Figure 5: Empirical CDF of the number of competitors (from the top 37 advertisers) per impression.

4 Machine Learning Framework

We start with the problem definition. To target ads, we need to develop a model that accurately predicts whether an impression will generate a click or not. Thus, our problem is one of CTR prediction. To solve this problem, we need to have three inputs (in addition to the data) – 1) Evaluation metric, 2) Feature set, and 3) Classifying algorithm. We discuss each of these below.

4.1 Evaluation Metric

To evaluate whether a model improves our ability to predict clicks, we first need to define a measure of predictive accuracy or an evaluation metric. There is no one ideal metric for all situations and the appropriateness of the metric depends on the goals of the project. In our case, we need a metric that can – 1) Accurately capture the improvement in CTR prediction for each impression (since the accuracy of this prediction has implications for the revenue and surplus of the platform as well as the advertisers). This is in contrast to classification tasks, where only the relative ordering of the predictions matters. 2) Be used to compare models with different features and evaluate the value of different types of information.

A metric that satisfies both these objectives is “Relative Information Gain” or *RIG*, and we use it as our primary evaluation metric. *RIG* is based on LogLoss, which is the most commonly used metric in the CTR prediction literature (Yi et al., 2013).

RIG is formally defined as follows. Consider a data set with N impressions, where $\mathbf{y} = \{y_1, y_2, \dots, y_N\}$ denotes the set of observed click indicator for the N impressions. For a given model A, let $\hat{\mathbf{y}}_\mathbf{A} = \{\hat{y}_{A1}, \hat{y}_{A2}, \dots, \hat{y}_{AN}\}$ denote the CTRs estimated by the model. Then the LogLoss of model $\mathbf{A}$ is given by:

$$\mathcal{L}^{log\ loss}(\hat{\mathbf{y}}_{\mathbf{A}}, \mathbf{y}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log\left(\hat{y}_{Ai}\right) + (1-y_i)\log\left(1-\hat{y}_{Ai}\right)\right) \tag{2}$$

By definition, LogLoss is the negative log of likelihood for our prediction model. Hence, the higher the LogLoss, the worse our model's performance. In order to compare the performance of different models, we define *RIG* as:

$$RIG(\hat{\mathbf{y}}_{\mathbf{A}}, \hat{\mathbf{y}}_{\mathbf{B}}; \mathbf{y}) = \left(1 - \frac{\mathcal{L}^{log\ loss}(\hat{\mathbf{y}}_{\mathbf{A}}, \mathbf{y})}{\mathcal{L}^{log\ loss}(\hat{\mathbf{y}}_{\mathbf{B}}, \mathbf{y})}\right) \times 100 \tag{3}$$

where $\hat{\mathbf{y}}_{\mathbf{A}}$ and $\hat{\mathbf{y}}_{\mathbf{B}}$ are the predictions from models A and B, respectively. *RIG* can thus be interpreted as the percentage improvement in LogLoss that we achieve by going from model B to model A. If the baseline model A performs better than model B, then *RIG* would be negative.

In our empirical analysis, we always use average observed CTR for the data as the baseline model (unless otherwise specified). This is the simplest aggregate metric that we can derive from the data, and it tells us how well we can do without any model. It is important to control for this baseline because if the baseline CTR in the data is very high (close to 1) or very low (close to zero, as in most e-commerce settings, including ours), a naive prediction based on average CTRs can provide a pretty good LogLoss. By defining *RIG* over this baseline, we are able to evaluate the relative benefit of a given model over and above a naive prediction. Moreover, normalizing the LogLoss with the average CTR reduces the sensitivity of the metric to the data distribution (He et al., 2014). Nevertheless, we need to be careful when interpreting *RIG*s computed on different datasets because there is no obvious normalization in those cases (Yi et al., 2013).

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

4.2 Feature Generation Framework

Feature generation is an important step in all machine learning problems. Since our goal is CTR prediction, we need a set of informative features that can help accurately predict the probability of click for a given impression.

We follow the functional feature generation framework proposed by Yoganarasimhan (2017). There are three advantages to adopting her framework. First, her function-based approach allows us to generate a large and varied set of features using a parsimonious series of functions instead of defining each feature individually. Second, it allows for a natural mapping between feature inputs and feature classification. Third, the general class of features she suggests have been shown to offer significant predictive power in these classes of problems.

4.2.1 Inputs for Feature Functions

Each impression i in our training, validation, and test data can be uniquely characterized by the following four variables – (1) the hour of the day during which the impression occurred (denoted by t_i), (2) the app p_i within which it occurred, (3) the user u_i who generated the impression, and (4) the ad a_i which was shown in the impression. 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: $\theta_{ia} \in \Theta_{ia} = \{a_i, \varnothing\}$. Here, a_i denotes the ad shown in i^{th} impression, where $a_i \in \mathcal{A} = \{a^{(1)}, a^{(2)}, \ldots, a^{(37)}, a^{(s)}\}$. The elements $a^{(1)}, a^{(2)}, \ldots, a^{(37)}$ refer to the top 37 ads, whereas all the smaller ads are grouped into one category, which is denoted by as $a^{(s)}$. Finally, if $\theta_{ia} = \varnothing$, 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 *app* and *time* related information for impression i. Together they capture the context (where and when) for the impression.

   (a) $\theta_{ip} \in \Theta_{ip} = \{p_i, \varnothing\}$ can take two possible values. $\theta_{ip} = p_i$ refers to the app where the impression was shown and $p_i \in \mathcal{P} = \{p^{(1)}, p^{(2)}, \ldots, p^{(50)}, p^{(s)}\}$. $p^{(1)}$ 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). Finally, $\theta_{ip} = \varnothing$ means that the function is aggregated over all apps.

   (b) $\theta_{it} \in \Theta_{it} = \{t_i, \varnothing\}$ characterizes the time-related information for impression i, where t_i denotes the hour of the day during which the impression occurred, and $\varnothing$ which implies that the function is aggregated over all hours of the day. Naturally, t_i can take 24 possible values ranging from 1 through 24.

3. *Behavioral information*: The fourth input $\theta_{iu} \in \Theta_{iu} = \{u_i, \varnothing\}$ captures the behavioral information for impression i. If $\theta_{iu} = u_i$, it means that the feature is specified for user u_i who

generated the impression. Features that use this information are the ones that can allow us to do behavioral targeting. In general, $u_i \in \mathcal{U} = \{u^{(1)}, \ldots, u^{(728{,}340)}\}$, where $\mathcal{U}$ is the full sample of users in the training, validation, and test data. As usual, $\theta_{iu} = \varnothing$ denotes that the function is aggregated over all possible 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 is available.
   - $\eta_{ib} = s$ denotes **s**hort-term history, i.e., only data on or after Oct 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.[10] 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 28 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 6, we present a picture with five example users to illustrate different types of history.

The domain for our feature functions is the Cartesian product of the sets described above. For example, $\Theta_i = \Theta_{ia} \times \Theta_{ip} \times \Theta_{it} \times \Theta_{iu} \times \mathcal{H}_{ib} \times \mathcal{H}_{ie}$ is the domain for functions that take all the inputs.

4.2.2 Feature Functions

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

[10]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 latter impression belongs to a different session than the former.

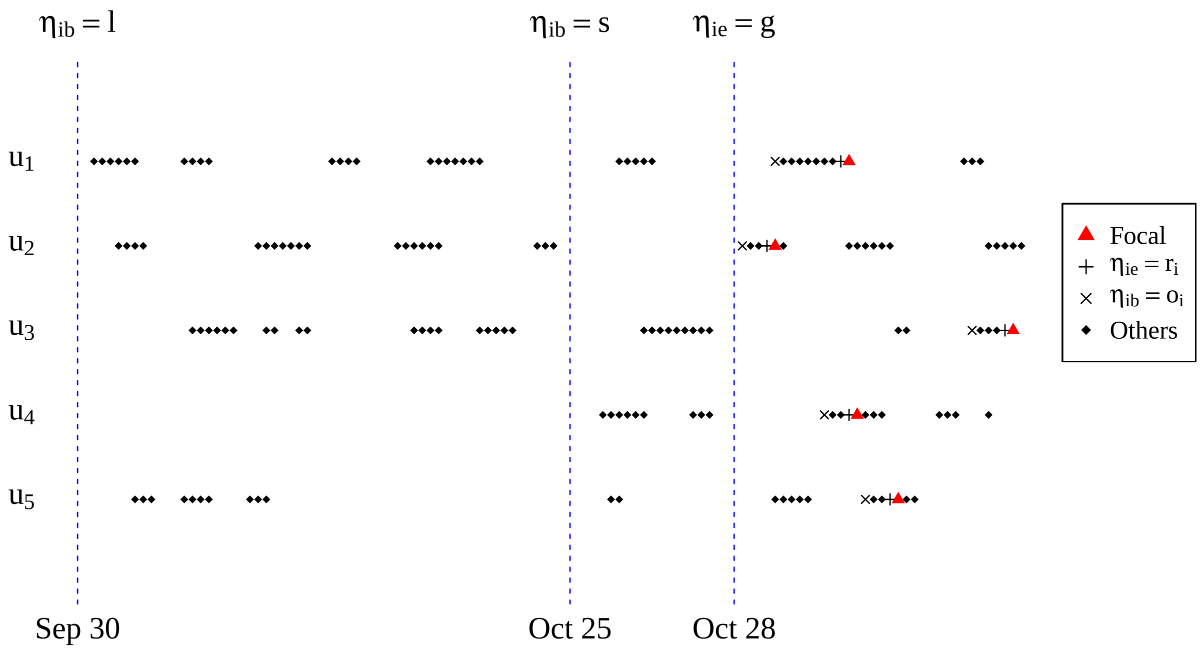


Figure 6: Depiction of history for five example users.The ▲ refers to the focal impression i for which the features are being generated. $+$ denotes the last impression just before the focal impression, and $\times$ refers to the first impression in the session in which the focal impression occurs.

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. Formally, $Impressions : \Theta_i \rightarrow \mathbb{Z}^+ \cup \{0\}$ takes an input from Θ_i and returns the number of impressions with the specified characteristics ($\mathbb{Z}^+$ represents the set of positive integers). Impressions is defined as follows:

$$Impressions(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \sum_{j \in [\eta_{ib}, \eta_{ie}]} \mathbb{1}(A_j = \theta_{ia})\mathbb{1}(P_j = \theta_{ip})\mathbb{1}(T_j = \theta_{it})\mathbb{1}(U_j = \theta_{iu}), \tag{4}$$

where U, A, P, and T are random variables indicating user, ad, app, and time respectively. The summation is over the two history variables – the starting point of the history η_{ib} and end point of the history η_{ie}. For example, $Impressions(a_i, p_i, t_i, u_i; l, r_i)$ returns the number of times ad a_i is shown to user u_i while using app a_i at the hour of day t_i in the time period starting September 30 2015 and ending with the last impression before i.

If instead 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 ($\eta_{ib} = s$) till the end of the global data ($\eta_{ie} = g$), we would have:

$$\begin{aligned}
Impressions(a_i, \varnothing, \varnothing, u_i; s, g) &= \sum_{j\in[s,g]} \mathbb{1}(A_j = a_i)\mathbb{1}(P_j = \varnothing)\mathbb{1}(T_j = \varnothing)\mathbb{1}(U_j = u_i) \\
&= \sum_{j\in[s,g]} \mathbb{1}(A_j = a_i)\mathbb{1}(U_j = u_i)
\end{aligned}$$

In general, the $Impressions$ function aims to capture the effects of repeated ad exposure on user behavior that has been shown to yield higher ad effectiveness in the recent literature (Sahni, 2015; Johnson et al., 2016). This function 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. The domain and range are the same as $Impressions$ function. The only difference is that $Clicks$ only counts the impressions that led to a click. Let C denote the binary random variable for a click. Then:

$$Clicks(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \sum_{j\in[\eta_{ib},\eta_{ie}]} \mathbb{1}(A_j = \theta_{ia})\mathbb{1}(P_j = \theta_{ip})\mathbb{1}(T_j = \theta_{it})\mathbb{1}(U_j = \theta_{iu})\mathbb{1}(C_j = 1) \tag{5}$$

$Clicks$ 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. Formally, $CTR : \Theta_i \to [0, 1]$, takes an input from Θ_i and returns CTR. If $Impressions(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) > 0$:

$$CTR(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \frac{Clicks(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie})}{Impressions(\theta_{ia}, \theta_{ip}, \theta_{it}, \theta_{iu}; \eta_{ib}, \eta_{ie})}, \tag{6}$$

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 $Clicks$. Intuitively, features generated based on this function capture the click-propensity at the user, ad, app, and time level and their interactions. It thus explains a substantial variation in the data.[11]

[11]In principle, we do not need to CTR over and above $Clicks$ and $Impressions$ if we our learning model can automatically generate new features based on non-linear combinations of basic features. Machine learning models like

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, *e.g.*, the number of distinct ads a user has seen while using a given app. Formally, $AdCount : \Theta_{ip} \times \Theta_{iu} \times \mathcal{H}_{ib} \times \mathcal{H}_{ie} \to \mathbb{Z}^+ \cup \{0\}$ is defined as:

$$AdCount(\theta_{ip}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \sum_{a \in \mathcal{A}^F} \mathbb{1}(Impressions(a, \theta_{ip}, \varnothing, \theta_{iu}; \eta_{ib}, \eta_{ie}) > 0) \tag{7}$$

   Features derived using this function capture the variety in ads seen by users within and across apps for different histories. Behavioral literature suggests that when consumers view ads with higher variety, they are more likely to engage with them (Redden, 2007).

   We use the full set of ads seen in our data (denoted by $\mathcal{A}^F$) and not just the top 37 ads to calculate this function.[12] This ensures that we are capturing the full extent of variety in ads seen by users.

5. $Entropy(\theta_{ip}, \theta_{iu}; \eta_{ib}, \eta_{ie})$: This function captures entropy or dispersion in the ads seen by a user. We use Simpson (1949)'s measure of diversity as our entropy metric. $Entropy$: $\Theta_{ip} \times \Theta_{iu} \times \mathcal{H}_{ib} \times \mathcal{H}_{ie} \to \mathbb{R}$ is defined as follows:

$$Entropy(\theta_{ip}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \frac{1}{\sum_{a \in \mathcal{A}^F} Impressions(a, \theta_{ip}, \varnothing, \theta_{iu}; \eta_{ib}, \eta_{ie})^2} \tag{8}$$

   $Entropy$ 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 consumers' 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 CTR prediction model.

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 model comparisons.

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

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. It is thus similar to $AdCount$. $AppCount : \Theta_{ia} \times \Theta_{iu} \times \mathcal{H}_{ib} \times \mathcal{H}_{ie} \to \mathbb{Z}^+ \cup \{0\}$ is defined as follows:

$$AppCount(\theta_{ia}, \theta_{iu}; \eta_{ib}, \eta_{ie}) = \sum_{p \in \mathcal{P}} \mathbb{1}(Impressions(\theta_{ia}, p, \varnothing, \theta_{iu}; \eta_{ib}, \eta_{ie}) > 0), \tag{9}$$

where $\mathcal{P}$ is the set of apps, as defined in §4.2.1.[13] 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 different apps. We also expect the click behavior of users to vary based on the number of apps they use. $AppCount$-based features help us capture these differences.

7. $TimeVariability(\theta_{iu}; \eta_{ib}, \eta_{ie})$: This function measures the variance in a user's CTR over different hours of the day. $TimeVariability : \Theta_{iu} \times \mathcal{H}_{ib} \times \mathcal{H}_{ie} \to \mathbb{R}$ is defined as follows:

$$TimeVariability(\theta_{iu}; \eta_{ib}, \eta_{ie}) = \text{Var}_t[CTR(\varnothing, \varnothing, t, \theta_{iu}; \eta_{ib}, \eta_{ie})] \tag{10}$$

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 $TimeVariability$. $AppVariability : \Theta_{iu} \times \mathcal{H}_{ib} \times \mathcal{H}_{ie} \to \mathbb{R}$ is defined as follows:

$$AppVariability(\theta_{iu}; \eta_{ib}, \eta_{ie}) = \text{Var}_p[CTR(\varnothing, p, \varnothing, \theta_{iu}; \eta_{ib}, \eta_{ie})] \tag{11}$$

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

In addition, we include the following standalone features in our model:

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.

[13] $\mathcal{P}$ 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.

12. $WiFi(u_i)$: Denotes whether the user is connected via WiFi 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.

4.2.3 Feature Classification

We now use functions defined above to generate a total of 161 features for each impression. We present the full list of features in Table A1 in Appendix §B.

To aid our analysis, we classify features based on the inputs used to generate them into the following (partially overlapping) categories:

- *Ad*-specific features (F_A): These are features that contain information on the ad shown during the impression. Formally, an ad-specific feature is one that is generated with input $\theta_{ia} \neq \varnothing$. All the features generated using the functions $AdCount$, $Entropy$, Bid, and $AdDummy$ are also included in F_A since they use ad-specific information.
- *Contextual* features (F_C): These are features that contain information on the context of the impression. Since our feature functions can take two types of contextual input, we further classify them into two (partially overlapping) sub-categories:
  - *App*-specific features (F_P): These features contain information on the app in which the impression was shown. Formally, an app-specific feature is one that is generated with input $\theta_{ip} \neq \varnothing$. It also includes all features generated using the functions $AppCount$ and $AppVariability$ since they both use app-level information.
  - *Time*-level specific features (F_T): These features contain information on the hour of the day during which the impression occurred. Formally, a time-specific feature is one that

is generated with input $\theta_{it} \neq \varnothing$. It also includes all features generated using the function $TimeVariability$.

- *Behavioral* features (F_B): These are features that contain information on the behavior of the user who generated the impression. Formally, a behavioral feature is one that is generated with input $\theta_{iu} \neq \varnothing$. In addition, features generated using $Lattitude$, $Longitude$, $WiFi$, $Brand$, MSP, ISP are also classified as behavioral since they contain user-specific information.

Together, these three feature sets form the full set of features $F_F = F_B \cup F_C \cup F_A$. All the feature classifications are shown in Table A1 and in §5.2, we examine the relative value of these different categories of features in their ability to predict clicks and improve targeting outcomes.

4.3 Learning using Boosted Regression Trees

For the prediction task, we use the Extreme Gradient Boosting with Regression Trees (XGBoost) algorithm proposed by Chen and Guestrin (2016). In §4.3.1 we give a brief overview of boosting and XGBoost. Then, in §4.3.2, we present a detailed exposition of the method.

4.3.1 Boosting

The concept of boosting was first introduced by Schapire (1990) and Freund (1995) who showed that it is possible to generate a strong learner using a combination of weak learners. Soon after, Freund and Schapire (1996) developed the first boosting algorithm, AdaBoost. In an important paper, Breiman (1998) showed that boosting can be interpreted as gradient descent in function space. This view was expanded by Friedman (2001), who showed how boosting can be applied to a large set of loss functions using any underlying weak learner (e.g., CART, logistic functions, regressions). Since then, boosted regression trees, also known as MART (multivariate adaptive regression trees) have been used to solve a variety of prediction problems.

MART can be viewed as performing gradient descent in the function space using shallow CART or regression trees (with a small number of leaves). CART is scalable, easy to interpret, can handle a mixture of discrete and continuous inputs, is insensitive to monotone transformations, and performs automatic variable selection (Murphy, 2012). However, it has limited accuracy because of its discontinuous nature and because it is trained using greedy algorithms. In particular, shallow regression trees tend to low variance, but high bias. Boosting allows us to preserve the low variance and reduce the bias. Thus, MART produces high quality classifiers that combine the positive aspects of CART with those of boosting (Caruana and Niculescu-Mizil, 2006).

More recently, Chen and Guestrin (2016) introduced a new boosted tree algorithm, XGBoost, that improves on MART. XGBoost differs from MART on four key dimensions. First, from a methodological standpoint, it can be interpreted as performing Newton boosting in the function space (as opposed to gradient descent), and thereby uses information from the Hessian as well. Thus, both the quality of the leaf structure and the leaf weights learned are more accurate in each step. Second, XGBoost uses a trick commonly used in Random Forests – column sub-sampling, which reduces the correlation between subsequent trees. Third, XGBoost employs a sparsity-aware split finding, which makes the algorithm run faster on sparse/missing data. Finally, from an implementation perspective, XGBoost is highly parallelized, which makes it fast and scalable.

XGBoost is one of the most successful prediction algorithms developed in the last few years. From 2015 onwards, most of the KDD cup winners have used XGBoost as their learning algorithm (either as a standalone model or in ensembles).[14] Further, Chen and Guestrin (2016) note that "Among the 29 challenge winning solutions published at Kaggle's blog during 2015, 17 solutions used XGBoost." In sum, boosted tree algorithms in general, and XGBoost in particular, have been shown to perform exceptionally well in tasks involving predicting human behavior.[15]

4.3.2 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:

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{j=1}^{J} \tau_j(\mathbf{x}_i) = \sum_{j=1}^{J} w^{(j)}_{q_j(\mathbf{x}_i)}, \tag{12}$$

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. 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:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \gamma \sum_j L_j + \frac{1}{2}\lambda \sum_j \left\| w^{(j)} \right\|^2, \tag{13}$$

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 § 4.1). Here, in contrast to MART, XGBoost penalizes not just tree depth but leaf weights as well.

Since the regularized objective in Equation (13) cannot be minimized using traditional optimiza-

[14]Ensemble methods outperform a well-configured boosted tree model by only a small amount (Bekkerman, 2015). This is important because ensemble models come with very high computational costs.

[15]Examples include store sales prediction, customer behavior prediction, product categorization, ad CTR prediction, course dropout rate prediction, etc.

tion 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:

$$\mathcal{L}(\phi) = \sum_i l\big(\hat{y}_i^{(j-1)} + \tau_j(\mathbf{x}_i), y_i\big) + \gamma L_j + \frac{1}{2}\lambda \left\| w^{(j)} \right\|^2 \tag{14}$$

In each iteration, we greedily add the tree that most improves our model according to the objective function in Equation (13). 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:

$$\mathcal{L}(\phi) \simeq \sum_i \left[l(\hat{y}_i^{(j-1)}, y_i) + \mathcal{G}_i \tau_j(\mathbf{x}_i) + \frac{1}{2}\mathcal{H}_i \tau_j^2(\mathbf{x}_i) \right] + \gamma L_j + \frac{1}{2}\lambda \left\| w^{(j)} \right\|^2, \tag{15}$$

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.

Note that implementation of this optimization routine requires the researcher to provide a set of hyper-parameters. These include the regularization parameters γ and λ defined in Equation 13. 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 let the algorithm end. 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 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. Appendix §A provides a step by step explanation and the final values used for these hyper-parameters in our analysis.

5 Results

This section is organized as follows. In §5.1, we discuss the gains in predictive ability from our machine learning framework. In §5.2, we present an analysis of how different classes of features

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

Table 3: LogLoss and *RIG* (in percentage) shown for training, validation, and test data.

contribute to these gains. In §5.3, we examine how stricter privacy regulations on user tracking would affect the ability to target. Finally, in §5.4 we discuss the robustness and scalability of our approach.

5.1 Gains in Targeting Ability

Table 3 shows the gains in predictive ability on both the training and validation data and the test data. The first row depicts the LogLoss for the full model (which uses the set of 161 features derived in §4.2 and the gradient boosting algorithm described in §4.3). The second row depicts the LogLoss for the baseline model, which is simply the average CTR for the dataset. The third row is the *RIG* of the full model compared to the baseline.

The *RIG* of the full model over the baseline is 17.95% on the test data. This improvement is quite substantial in CTR prediction problems (He et al., 2014). It suggests that our machine learning framework significantly improves our ability to predict whether an impression will receive a click or not. From the ad network and advertisers' perspectives, this improvement directly translates to improved ability to target, or provide a better match between impressions and ads. Further, in §5.4 and Appendix §C.1, we show that the improvements in predictive ability are similar when we use other evaluation metrics such as AUC, percentage improvement in MSE, and 0/1 Loss.

The *RIG* improvement for training and validation data is 18.47%, which is somewhat higher than 17.95% for the test data. There are two potential reasons for this. First, all statistical models estimated on finite data have higher in-sample fit than out-of-sample fit. Indeed, this is the main reason we use the test data to evaluate model performance. Second, the difference could simply reflect the differences in the underlying data distributions for the two data-sets. As discussed in §4.1, we cannot compare *RIG* across data-sets because it is co-determined by both the model and the data. Thus, the difference between the *RIG* values across the data-sets is not necessarily informative.

5.2 Value of Feature Sets

We now examine the impact of different types of features on the predictive accuracy of our model. This is important for three reasons. First, it gives us intuition on which consumers click, how they click, when they click, and what types of ads they click on. Second, data storage and processing costs vary across feature types. For example, some user-specific behavioral features require real-

RIG of A over B	**Full Sample**	**Top Ads and Top Apps**
Behavioral over Baseline	12.14%	14.82%
Contextual over Baseline	5.25%	5.98%
Full over Baseline	17.95%	22.85%
Behavioral over Contextual	7.27%	9.40%
No. of Impressions	9,625,835	6,108,511
Percentage of Test Data	100%	63.5%

Table 4: Comparison of Behavioral and Contextual models for different samples of test data.

time updating, whereas pure-contextual features tend to be more stable, and can be updated less frequently. In order to decide whether to store and update a feature or not, we need to know its incremental value in improving targeting. Third, the privacy and policy implications of targeting depend on the features used. For example, models that use behavioral features are less privacy-preserving than those that use purely contextual features. Before adopting models that are weaker on privacy, we need objective measures of whether such models actually perform better.

We use the feature categorization discussed in §4.2.3 for this analysis. Recall that we categorized features into three broad overlapping sets – 1) Behavioral, denoted by F_B, 2) Contextual, denoted by F_C, and 3) Ad-specific, denoted by F_A. Within contextual, features are further sub-categorized as App-specific (F_P) and Time-specific features (F_T).

5.2.1 Behavioral vs. Contextual Features

To evaluate the relative value of behavioral and contextual information, we define two models:

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

Both models include ad-specific features that are neither behavioral nor contextual (e.g., Feature 2 in Table A1) because it is reasonable to expect the ad-network to have this information.[16] They also use the same objective and training algorithm and only differ on the set of features used. Hence, it is possible for us to directly compare the *RIG* of one model over another within the same data.[17]

The results from these two models and their comparisons with the Baseline model are presented

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

[17]As discussed in §4.1, *RIG*s are not directly comparable across different data-sets. Simply put, in Table 4 comparisons within a column are interpret-able, but comparisons across a row are not.

in Table 4. First, consider the results for the full test data (presented in the second column). The Behavioral model has a 12.27% *RIG* over the baseline, which is considerably higher than 5.12% of the Contextual model over the baseline.[18] Directly comparing the Behavioral model to the Contextual model gives us an *RIG* of 7.54%. Together, these findings suggest that behavioral targeting is more effective compared to contextual targeting in mobile in-app advertising. While it is possible that additional contextual information (that we do not have) can change the relative ordering of these findings, our findings establish a base case for these comparative results.

One possible critique of the above analysis is that it does not exploit the full capacity of contextual information since we pool the data for the non-top ads and non-top apps during feature generation. For example, all small ads are treated as one ad and simply identified as $\theta_{ia} = a^{(s)}$, which makes the contextual information murky. To address this issue, we consider a sub-sample of the test data which only consists of impressions that were shown in a top app and showed a top ad and re-run all the above comparisons. This accounts for 63.5% of our test data. The performance of our Full model on this subset of the data is even better than that on the full sample because there is no information loss on the ads or apps. The findings on the relative value of behavioral vs. contextual features are even stronger in this data-set, which suggests that our results in the full sample were not driven by the lack of good contextual information.

5.2.2 Incremental Value of Different Types of Features

Having established the relative value of behavioral and contextual information, we now delve deeper into the value of different feature sets. We quantify the relative value of each of the following feature sets – ad-specific features (F_A), app-specific features (F_P), time (F_T), and behavioral features (F_B). For each feature-set, we consider the following two models:

- **Forward Addition Model:** This model starts with the null set and then adds features generated with a particular piece of information alone. Forward addition models demonstrate how much we can gain by using only one piece of information.
- **Backward Deduction Model:** This model starts with the full set of features and then deducts features generated using a particular piece of information. Backward deduction models show the loss in predictive accuracy when we do not have a specific piece of information.

Table 5 presents the results for both these models for four types of features.

Ad information: The forward addition model that only includes ad-related information reaches an *RIG* of 0.99% over the baseline, i.e., the gain from purely ad-related features is quite small. This is also borne out by the fact that the *RIG* of the backward deduction model for ad-features is 17.26%,

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

Piece of Information	**Forward Addition Model**		**Backward Deduction Model**	
	Feature Set	RIG	Feature Set	RIG
Ad	$F_A \setminus (F_P \cup F_T \cup F_B)$	0.99%	$F_F \setminus F_A$	17.26%
App	$F_P \setminus (F_A \cup F_T \cup F_B)$	4.39%	$F_F \setminus F_P$	15.08%
Time	$F_T \setminus (F_A \cup F_P \cup F_B)$	0.11%	$F_F \setminus F_T$	15.65%
Behavioral	$F_B \setminus (F_A \cup F_P \cup F_T)$	10.26%	$F_F \setminus F_B$	5.25%

Table 5: RIG of Model with/without Different Pieces of Information over Baseline CTR

which is very close to 17.95%, the *RIG* of the full model (see Table 4).

Together, these results imply that ad information adds only marginally to the predictive accuracy of our framework. This finding likely stems from three reasons. First, it could be due to the fact that in-app ads are quite small and cannot convey much information or have significant persuasive quality. Second, since we do not have access to the creatives used, we do not have detailed ad-specific features such as message or design that can influence clicks. Finally, because all ads are shown to all users due to the randomization from the proportional auction, there is no additional information on user-segments in the ad-specific features.

App information: The forward addition model that includes only app-level features has an *RIG* of 4.39% over the baseline, while the backward deduction model that subtracts app-specific features from the full model has an *RIG* of 15.07% over the baseline. The value of app-level features is thus higher than that of ad-level features.

Users self-select into apps, which in turn makes app-level features indicative of underlying user-segments. Further, aggregated app-specific features may capture both user interface (UI) and user experience (UX) factors that affect users' propensity to click within a particular app. Both these reasons may play an important role in predicting clicks.

Time information: The forward addition model with time information has an *RIG* of 0.11%, which is minimal. However, the backward deduction model without time-specific features has an *RIG* of 15.65% over the baseline. This translates to a relative information loss of 2.80% compared to the full model, which is quite high.[19] This suggests that although temporal information alone does not help our prediction model much, its interactions with other pieces of information are valuable.

Behavioral information: Finally, we examine the value of behavioral features. This analysis is analogous to the comparison of Behavioral and Contextual models presented in §5.2.1. The forward addition model with behavioral features is the Behavioral model excluding ad-specific features, while backward deduction model without behavioral features is identical to the Contextual model.

[19]We define relative information loss as the percentage loss in LogLoss compared to the full model.

Interestingly, we find that the forward addition model performs better than the backward deduction model without behavioral information. This result is in stark contrast to the results for other pieces of information, and it reiterates the unique value of behavioral features in predicting CTR.

5.3 Targetability and User Identifiers

User-identification and tracking are at the core of targeting technology in the digital advertising industry. Mobile in-app advertisements rely on a user re-settable device ID for tracking purposes (referred to as AAID in our data). Privacy advocates have demanded that mobile phone makers disable such tracking, whereas advertisers have argued that this would lead to significant loss in their targetability (and hence revenues). As discussed earlier, Apple has already taken a step in this direction with the introduction of LAT.

Consider a situation where consumer privacy laws are strengthened so as prevent the use of a tracking identifier (such as the broad implementation of LAT). Under this new regime, the platform and advertisers would have to rely on IP as their user-identifier. While IP address is a commonly-used tracking metric in online settings, it is problematic for two reasons. First, all users behind the same NAT firewall or proxy have the same external IP address. So when we use IP address as the user-identifier, all of them are grouped under the same ID and identified as a single user. Second, IP addresses are generally not static, especially in the case of mobile phones. When a user switches from a WiFi connection to 3G/4G (or vice-versa), the IP address changes. Thus, the same user will show up with different IP addresses in the data.

To examine how using IPs would affect the platform's targeting ability, we re-do all our analysis using IP as our user-identifier instead of AAID. The sampling, feature-generation, training, and testing are all analogous to the earlier analysis, except now IP is used as the user ID. We first draw a sample of 799,219 unique IPs from October 28 to October 30 and then track them back to September 30. This gives us 162,869,975 impressions in global data, 19,574,051 impressions in the training and validation data, and 10,614,219 in the test data.[20] We then generate all the features for the impressions in the training, validation, and test data-sets based on this data. Note that because IP is now our user-identifier, all user-specific behavioral features are now defined over the same IP instead of the same AAID. We then train model and test its performance on the test data. The results are presented in Table 6. There are major differences with Table 4 that are worth noting. First, the overall performance of the model drops substantially – from 17.95% to 13.38%. This indicates that there is considerable information loss when we move from AAID to IP. Second, we find that the Contextual model performs slightly better than the Behavioral model. With full sample, the RIG

[20]The sample size is chosen to be similar to the one used in the main analysis to enable us to compare results from this section with the main analysis.

RIG	**Full Sample**	**Top Ads/Apps**	**Users with Long History**
Behavioral over Baseline	5.04%	5.52%	3.47%
Contextual over Baseline	5.33%	6.28%	6.43%
Full over Baseline	13.38%	16.64%	10.56%
Contextual over Behavioral	0.31%	0.80%	3.07%
No. of Impressions	10,614,219	6,751,331	1,711,613
Percentage of Test Data	100%	63.61%	16.13%

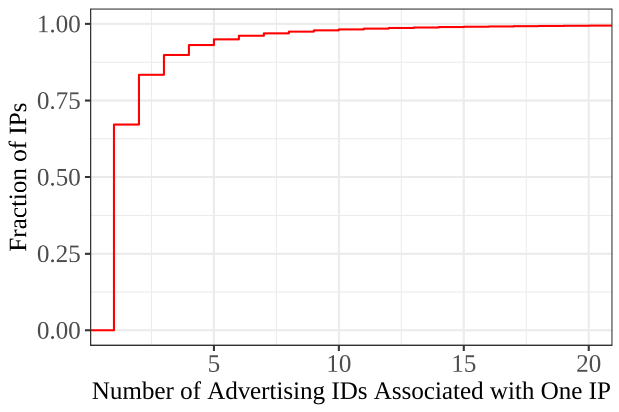
Table 6: *RIG* for different model specifications and samples using IP as the user-identifier.

of the Contextual model over the Behavioral model is 0.31% and when we constrain the data to top ads and top apps, this number increases to 0.80%. Again, this indicates that there is significant information loss at user-level features with IP as the identifier.

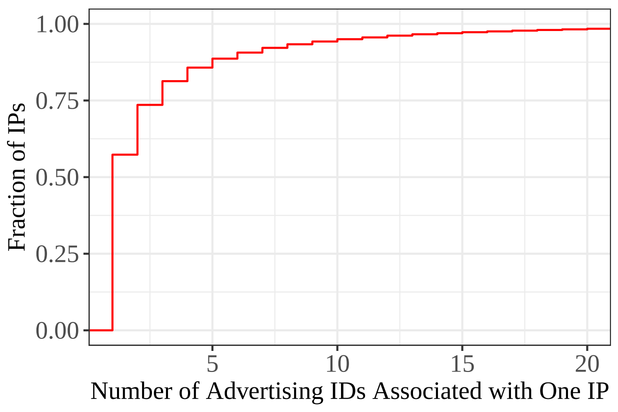
We now examine if the loss in behavioral information is mitigated if we focus on users with long history. So we present the results for impressions where the user (as identified by the IP) has at least seen 1000 impressions from September 30 to October 29. Surprisingly, the Contextual model outperforms the Behavioral model in this case, even more than the other samples; the *RIG* of the Contextual model over the Behavioral model is 3.07%. We believe this stems from one of the fundamental problems with using IP as a user-identifier: an IP address with long history is actually a combination of different users with the same IP. Figure 7 presents a visual illustration of this problem. In the full test data, 33% of unique IPs corresponds to two or more AAIDs (see Figure 7a). This percentage rises to 43% when we only focus on users addresses with long history (see Figure 7b). Thus, as we track users longer, the amount of noise in the tracking (and hence behavioral features) increases. Thus, tracking users for longer leads to worse targeting when we use a noisy identifier such as IP.[21]

In sum, there are two problems with using IP: 1) we treat different users as one, 2) we may observe different IPs for one user. As such, the accumulation of user history is not be very helpful when using IPs. However, only the latter is an issue with AAID, i.e., the same AAID can never correspond to multiple users. Thus, the noise in behavioral information with AAIDs is lower. Privacy regulations on tracking users can lead to either of these two problems. Our findings suggest that the former is a more important issue and that advertisers and policy-makers should be aware of the role of different sources of noise in user-identifiers.

[21] Another possibility is that users with the same IP and different AAIDs are the ones who re-set their AAID. However, we do not think this is the cause of the discrepancy between AAIDs and IPs for two reasons. First, practically not many users re-set their AAIDs regularly. Second, if IPs were indeed mapping to unique users, the performance of the Behavioral model would not be worse than that in the main analysis.



(a) Full sample



(b) Users with long history

Figure 7: Empirical CDF of number of AAIDs associated with one unique IP.

5.4 Robustness Checks

To confirm the robustness of our findings, we conduct checks on all three aspects of our ML approach – the evaluation metric used, the learning model used, the feature generation approach, as well as on the size of the data used. We discuss these tests briefly here and refer readers to Appendix §C for details.

1. First, we consider the robustness of our results to the evaluation metric used. We consider three alternative metrics – Area Under the ROC Curve (AUC), 0/1 Loss, and Mean-Squared Error (MSE). All the key points that we made earlier continue to hold when we use one of these other metrics. We refer readers to Appendix §C.1 for a detailed discussion of these metrics, and to Tables A2, A3, A4 for results using these alternative metrics (which are analogous to Tables 4 and 5 presented earlier).

2. Second, we examine whether other learning models can give us better predictive power than XGBoost. We consider five other commonly used learning models in different models – 1) Least Squares, 2) LASSO, 3) Logistic Regression, 4) CART (Classification And Regression Tree), and 5) Random Forests. We train, tune, and validate all the models using the same training and validation data. The performance of XGBoost is significantly better than that of the other methods; please see Table A5 in the Appendix §C.2 for the performance of these other models on the test data. Since we use the same data and features to train and validate all these models, the differences stem from their ability to optimize the objective function. Please see Appendix §C.2 for a discussion of these alternative learning models and the details on tuning and validation of these models.

3. Third, we run a few 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 of implementation 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 have the length of history fixed, instead of having the η_{ib} fixed 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 Oct 28 is the same time on Sep 30, while l_i for impression i on Oct 30 is the same time on Oct 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***.

4. Fourth, we present some checks to establish that our data sample is sufficient and large enough to produce reliable results. Recall that we follow a user-based sampling procedure, wherein we randomly select 728,340 users and track them for a month (which translates to 584,029 users in training+validation and 433,287 users in test data). In Table A6 in Appendix §C.3, we present the ***RIG*** gains for different sample sizes, starting with 1000 users. We find that the ***RIG*** gains start stabilizing with the sample of 100,000 users. After this point, sampling more users leads to only small improvements in the model performance. This suggests that our sample of 728,340 users is sufficient for our purposes.

5. Fifth, we examine whether our results are sensitive to the validation procedure used to pick the tuning parameters (described in Appendix §A). We consider two other approaches that are widely used in practice: 1) hold-out validation, and 2) k-fold cross validation. The details of each are presented in Appendix C.4. We find that our approach yields slightly higher predictive accuracy than other validation methods.

6. Finally, we show that our findings on using IP as the user-identifier are not driven by the fact that the data-sets used for the main analysis and the IP analysis came through two different sampling strategies. In Appendix §C.5, we run a series of tests on the 15% of impressions that included in both samples and show that the substantive findings on the unreliability of using IP as a user-identifier continue to hold.

6 Implications for Data-sharing and Targeting

We now use our machine learning model to examine the implications of changing the platform's data-sharing strategies. We focus on two important issues:

- To what extent is the platform incentivized to share data with advertisers and to enable micro-targeting? Is there an optimal data-sharing (and corresponding targeting) strategy from the platform's perspective?
- How does the total surplus accrued by advertisers vary across data-sharing scenarios? Are the incentives of all the advertisers aligned or is there heterogeneity in advertisers' preferences on the optimal level of data-sharing?

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

The rest of this section proceeds as follows. In §6.1, we present a simple example to fix ideas and highlight the platform's efficiency-revenue trade-off. In §6.2, we present a stylized model to characterize the platform's revenue under different data-sharing strategies. In §6.3, we combine this analytical model and the machine learning model developed earlier to derive empirical estimates of the platform's profit, advertisers' revenues, and total surplus under different counterfactual data-sharing strategies. Finally, in §6.4, we present the findings from our counterfactual analysis, the limitations of our approach, and conclude with a discussion on optimal mechanism design.

6.1 A Simple Example

Starting with Levin and Milgrom (2010), a growing body of theoretical literature argues that sharing too much targeting information with advertisers can thin auction markets which in turn would soften competition and make the platform worse. Please see §2 for a detailed discussion of this literature.

We now present a simple example to highlight this idea. Consider a platform with two advertisers ($a^{(1)}$ and $a^{(2)}$) competing for two impressions by two users ($u^{(1)}$ and $u^{(2)}$). There are two possible pricing mechanisms – Cost per Impression (CPI) and Cost Per Click (CPC). We can formally show that both CPI and CPC mechanisms generate the same revenues for the platform and advertisers under different targeting strategies, though the interpretations are slightly different. To avoid repetition, we focus on the CPI case throughout the text and refer readers to Appendix E for the CPC example and analysis.

We consider second price CPI auctions, where the highest bidder wins the impression and pays the bid of the second-highest bidder for the impression. These auctions have the useful property of truthful bidding by advertisers (Vickrey, 1961).[22] Further assume that the advertisers are symmetric in their valuation of a click and asymmetric in their match values for impressions (their valuation of a click is normalized to 1 hereafter). Equation (16) shows the match values between the advertisers and users, which can also 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 $u^{(2)}$.

$$eCTR = \begin{array}{c} \\ a^{(1)} \\ \\ a^{(2)} \end{array} \begin{array}{cc} u^{(1)} & u^{(2)} \\ \end{array} \begin{pmatrix} 0.5 & 0.1 \\ & \\ 0.1 & 0.3 \end{pmatrix} \longrightarrow \overset{\bar{u}}{\begin{pmatrix} 0.3 \\ \\ 0.2 \end{pmatrix}} \tag{16}$$

We now 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 scenarios are laid out in Table 7 and discussed below:

- No data disclosure – Here advertisers only know their aggregate match value over both users. So $a^{(1)}$ and $a^{(2)}$'s expected match over the two users are 0.3 and 0.2. In a second price auction, advertisers simply bid their expected valuations. So $a^{(1)}$ wins both impressions and pays the next highest bid, $b_{21} = b_{22} = 0.2$ for each impression. Therefore, the total revenue of platform is $R = 0.4$, and that of the advertisers 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 7. 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

[22]There exist other deterministic auction mechanisms (e.g., first-price auction) that are revenue equivalent to the second-price auction. For the sake of simplicity, we therefore focus on the second-price auction in our analysis, but the results are generalizable to other auction mechanisms.

No Data Disclosure (or No Targeting)	Full Data Disclosure (or Perfect Targeting)
For both impressions: Bids: Advertiser $a^{(1)}$: $b_{11} = b_{21} = 0.3$ Advertiser $a^{(2)}$: $b_{12} = b_{22} = 0.2$ Outcome: Advertiser $a^{(1)}$ wins both impressions and pays 0.2 per impression	For User $u^{(1)}$'s impression: Bids: Advertiser $a^{(1)}$: $b_{11} = 0.5$, Advertiser $a^{(2)}$: $b_{12} = 0.1$ Outcome: Advertiser $a^{(1)}$ wins $u^{(1)}$'s impression and pays 0.1 For User $u^{(2)}$'s impression: Bids: Advertiser $a^{(1)}$: $b_{11} = 0.1$, Advertiser $a^{(2)}$: $b_{12} = 0.3$ Outcome: Advertiser $a^{(2)}$ wins $u^{(2)}$'s impression and pays 0.1
Platform's expected revenue: $R = 2 \times 0.2 = 0.4$	Platform's expected revenue: $R = 0.1 + 0.1 = 0.2$
Advertiser's expected surplus: $W_1 = 2 \times (0.3 - 0.2) = 0.2$ $W_2 = 0$	Advertiser's expected surplus: $W_1 = 0.5 - 0.1 = 0.4$ $W_2 = 0.3 - 0.1 = 0.2$
Total expected surplus: $S = 0.6$	Total expected surplus: $S = 0.8$

Table 7: Example depicting two scenarios: 1) No data disclosure and 2) Full disclosure.

ensures higher advertiser revenues, with $W_1 = 0.4$ and $W_2 = 0.2$. However, the platform's revenue is now lower, with $R = 0.2$. Thus, even though ads are matched more efficiently and the total surplus generated is higher, the platform extracts less revenue.

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

6.2 Stylized Model

We now develop a simple analytical model that captures the relationship between the platform's data-disclosure strategy and the revenues of the platform and advertisers. As before, we assume that the platform uses a second-price auction with CPI pricing and that advertisers are symmetric in their valuation for a click (normalized to 1).

Consider a platform that receives I impressions and serves A advertisers. Let m_{ia} denote the match value (or eCTR) of ad a for impression i. Let M denote the match value matrix:

$$M = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1A} \\ m_{21} & m_{22} & \dots & m_{2A} \\ \vdots & \vdots & \ddots & \vdots \\ m_{I1} & m_{I2} & \dots & m_{IA} \end{bmatrix} \tag{17}$$

The expected value of an impression for an advertiser when they have perfect targeting information is simply its match value of that impression. Thus, if the platform reveals M to advertisers, they place targeted bids for each impression i equal to $b_{ia} = m_{ia}$. Each impression i is then sold to the advertiser with the highest match value (or bid) which is given by $\max_a m_{ia}$. The total surplus generated by the platform is:

$$S_P = \sum_{i=1}^{I} \max_a m_{ia} \tag{18}$$

In contrast, if the platform conceals information and only provides advertisers with their aggregate match values over all impressions ($\bar{m}_a = \frac{\sum_{i=1}^{I} m_{ia}}{I}$), then all advertisers place non-targeted bids over all impressions where $b_a = \bar{m}_a$. In this case, the advertiser with highest match value aggregated over all impressions wins all impressions and the total surplus generated by the platform is:

$$S_0 = \max_a \sum_{i=1}^{I} m_{ia} \tag{19}$$

We can easily show that $S_P \geq S_0$, i.e., perfect targeting will increase the total surplus generated in the market.

However, it is not clear how this surplus is distributed between advertisers and the platform. Under the no targeting strategy, the platform's revenue for each impression is the same and is equal to the second highest match value aggregated over all impressions. Defining $a_0^{(1)}$ as the winning ad under no targeting, the winner then pays $\max_{a \backslash a_0^{(1)}} \frac{1}{I} \sum_{i=1}^{I} m_{ia}$ per impression. Thus, the platform's revenue is:

$$R_0 = \max_{a \backslash a_0^{(1)}} \sum_{i=1}^{I} m_{ia} \tag{20}$$

In the case of perfect targeting, for a given impression i, the advertiser with the highest match value (or bid) pays the second highest match value (or bid). This is equal to $\max_{a \backslash a_{P,i}^{(1)}} m_{ia}$, where $a_{P,i}^{(1)}$ denotes the winning ad in i-th impression under perfect targeting. Then the platform's total revenue is:

$$R_P = \sum_{i=1}^{I} \max_{a \backslash a^{(1)}_{P,i}} m_{ia} \tag{21}$$

While we can theoretically show that $S_P \geq S_0$, there is no theoretical relationship between R_P and R_0. Thus, whether the platform's revenue increases or decreases with targeting is an empirical question.

6.3 Empirical Analysis of Targeting and Data Sharing

In line with our main results, we consider four different types of data-sharing arrangements:

1. No data-sharing – Advertisers only know their average CTR on the platform. So they cannot place targeted bids; their bid for all the impressions on the platform is the same.

2. Contextual data-sharing – The platform shares contextual data and outcomes (winner of impression and click indicator) for all impressions with all the advertisers. Advertisers are not given any behavioral/user data. Thus, they can generate all contextual and ad-specific features which are not behavioral, i.e., $(F_C \cup F_A) \setminus F_B$. Using these features, they can calibrate the Contextual model shown in §5.2.1. In this case, each advertiser can target its bid based on the context. However, advertisers cannot distinguish two different users in the same context, so a given advertiser submits the same bid for all users in the same context.

3. Behavioral data-sharing – The platform shares impression-level behavioral data and outcomes (winner of impression and click indicator) with all the advertisers. In this case advertisers can generate all behavioral and ad-specific features which are not contextual, i.e., $(F_B \cup F_A) \setminus F_C$. Using these features they can calibrate the Behavioral model shown in §5.2.1. In this case, advertisers can identify the user, but not the context. Thus, the optimal bid of each advertiser is user-specific, but not context-specific.

4. Full data-sharing – The platform share all the data with advertisers. In this case the advertisers can generate the full set of features F_F and calibrate the Full model shown in §5.1. Here, advertisers can identify both the context and user, and they can target their bids at the impression-level.

In each of these data-sharing arrangements, advertisers first calibrate a click-prediction model using the data available to them. Then whenever an impression arrives, they use their model to predict their match value for the impression and use that as their targeted bid. We briefly explain our empirical strategy for generating counterfactual estimates of the platform's profits and total surplus below and refer interested readers to Appendix §D for a step by step procedure.

Formally, let $\hat{M}^{\mathcal{N}}$, $\hat{M}^{\mathcal{C}}$, $\hat{M}^{\mathcal{B}}$, and $\hat{M}^{\mathcal{F}}$ denote the match-value matrix for the No data-sharing, Contextual, Behavioral, and Full models, respectively. In the stylized analytical model, all impressions always had A advertisers competing for it. However, empirically, we need to derive the appropriate set of ads competing for each individual impression i. As discussed in §3.5, naively assuming that all ads compete for all impressions can lead to problems because the accuracy of our targeting models is predicated on the assumption that the joint distribution of co-variates and clicks is the same in the test and training/validation data-sets. While this is true in observed test data, it may not always be true when we consider counterfactual targeting strategies. So we develop a filtering approach to derive the appropriate set of ads in the consideration set for each impression. Please see Appendix §D for details on the filtering approach.

Next, we define the arrays in matrices associated with the highest and second-highest match values as follows:

$$X^{\mathcal{T}}_{(1)} = \{(i,j) \mid \hat{m}^{\mathcal{T}}_{ij} \geq \hat{m}^{\mathcal{T}}_{ik} \,,\ \forall\, k\} \tag{22}$$

$$X^{\mathcal{T}}_{(2)} = \{(i,j) \mid \hat{m}^{\mathcal{T}}_{ij} \geq \hat{m}^{\mathcal{T}}_{ik} \,,\ \forall\, k \neq \underset{l}{\operatorname{argmax}}\, \hat{M}^{\mathcal{T}}_{il}\}, \tag{23}$$

where $\mathcal{T} \in \{\mathcal{N}, \mathcal{C}, \mathcal{B}, \mathcal{F}\}$. Then, the platform's revenue and the total surplus in the system for a data-sharing strategy $\mathcal{T}$ are:

$$\hat{R}^{\mathcal{T}} = \sum_{(i,j)\in X^{\mathcal{T}}_{(2)}} \hat{m}^{\mathcal{T}}_{ij} \quad \text{and} \quad \hat{S}^{\mathcal{T}} = \sum_{(i,j)\in X^{\mathcal{T}}_{(1)}} \hat{m}^{\mathcal{F}}_{ij} \tag{24}$$

Note that we use $\hat{M}^{\mathcal{F}}$ for surplus calculation, since it is the most accurate estimation of CTR at the impression-level that we have. Further, we define the advertisers' surplus under scenario $\mathcal{T}$ as the difference $\hat{W}^{\mathcal{T}} = \hat{S}^{\mathcal{T}} - \hat{R}^{\mathcal{T}}$.[23]

6.4 Counterfactual Results

The results from our counterfactual exercise are shown in Table 8.

6.4.1 Platform's Revenue

As our theory model predicts, more granular information sharing leads to higher efficiency in the market: the total surplus under full information sharing is 10.8% higher than the no data-sharing case (see Table 8). However, platform revenues exhibit more of an inverted U-shaped curve. They

[23] We can also interpret data-sharing as match value sharing, i.e., in scenario $\mathcal{T}$, the platform simply shares the vector of its match values, $\{\hat{m}^{\mathcal{T}}_{1a}, \hat{m}^{\mathcal{T}}_{2a}, \ldots, \hat{m}^{\mathcal{T}}_{ia}, \ldots, \hat{m}^{\mathcal{T}}_{Ia}\}$, with advertiser a. However, this interpretation is predicated on the assumption that the platform will behave truthfully, i.e., the platform will not shade the match values. This can only happen when the advertisers' and platform's incentives are fully aligned, which as we will see later is not always true.

Data-sharing Scenario	Total Surplus	Platform Revenue	Advertisers' Surplus
Full	0.01440	0.01201	0.00239
Behavioral	0.01390	0.01198	0.00192
Contextual	0.01347	0.01207	0.00140
No data-sharing	0.01299	0.01183	0.00116

Table 8: Platform revenues, advertisers' surplus, and total surplus for different levels of data-sharing scenarios. The numbers are reported in average terms per impression.

are maximized when the platform restricts data-sharing to the contextual level. When the platform shares behavioral information with advertisers, advertisers achieve a greater ability to target. While this increases the efficiency and total surplus of the system, much of this surplus is appropriated by advertisers and the platform's revenue suffers. Nevertheless, our findings are weaker than those predicted by theory models, i.e., while revenues reduce with more granular targeting, the drop is not very large. This suggests that the strong distributional assumptions on the match values in theory papers may not hold in real ad auctions.

Thus, the incentives of the platform are not perfectly aligned with that of the advertisers. However, the platform's optimal data-sharing strategy is privacy preserving and aligned with consumers' preferences. Our findings thus support the advertising industry's claim that external regulation is not necessary to reduce user-tracking/targeting, and that the industry can self-regulate.

Finally, our results should be interpreted cautiously with the necessary caveats. First, our assumption that all advertisers have symmetric valuations may not always hold. Ideally, we should derive distribution of advertisers' valuations from data and then use it. However, to do so, we would first need to analytically derive bidders' strategies in proportional auctions and then to take the analytical model to data to structurally estimate the distributions of advertisers' valuations. While this is possible in principle, it is beyond the scope of this work. Second, when estimating the revenues and surpluses under different targeting strategies, we use the same distribution of features as seen in the current data. However, one may expect that the joint distribution of features observed in the data would change in response to the targeting strategy because the impressions shown to users and their click behavior is a function of targeting. Thus, the platform's surplus and the machine learning targeting models can both co-evolve in the long run equilibrium. As such, we do not take a strong stance on the optimal level of targeting in long-run equilibrium. Rather, our goal is to provide some empirical evidence in support of the idea that excessive data-sharing and targeting may not be optimal for the platform.

Our findings raise many interesting questions on optimal mechanism design for mobile ad auctions. Limiting information-sharing to contextual data is an obvious strategy. However, this approach also reduces the total surplus and hence caps the platform's revenues. Thus, the optimal path for the platform may not be to restrict data-sharing, but instead to consider mechanisms that can do both – 1) increase efficiency and 2) extract the revenue from winning advertisers by shrinking the informational rent. For instance, the platform could share granular data, and also adopt the standard theoretical solution proposed for revenue extraction – optimal reserve prices (Myerson, 1981; Riley and Samuelson, 1981). Ostrovsky and Schwarz (2016) validate these theoretical predictions using field experiments for search ads. However, they only consider optimal reserve prices for broad sets of keywords and assume eCTRs to be homogeneous across advertisers. In contrast, we have a setting where each impression is a unique product and advertisers' match values for an impression are heterogeneous. To extract the maximum revenue, the platform has to set dynamic impression-specific reserve prices, a mechanism that the literature has not explored so far. Platforms can also consider different auction mechanisms. Celis et al. (2014) show that ad auctions often fail Myerson's conditions for optimality and provide some evidence that randomized auctions can lead to better platform revenues. Finally, the platform can charge advertisers for data, i.e., the platform could act as both an ad-network and data intermediary, and thereby extract the extra surplus generated from more efficient matching. Pancras and Sudhir (2007) focus on equilibrium effects and study this problem for price-targeting. The design and implementation of such data-sharing contracts for ad-targeting could a fruitful avenue of research and also have big implications for the ad industry and consumer privacy advocates (to what extent should firms be allowed to sell users' data).

6.4.2 Advertiser's Surplus

Now we examine advertisers' surplus under different data-sharing scenarios and examine how the total surplus is distributed among different ads. We begin by comparing the total surplus over all advertisers. As shown in Table 8, advertiser's surplus is increasing with more granular information sharing. This validates our theoretical prediction that more granular information helps advertisers by allowing them to generate more accurate estimates of their match values and place targeted bids. Under full information sharing, we find over 100% increase in advertisers' surplus compared to the baseline. Further, the incremental increase in advertisers' surplus from contextual to behavioral data-sharing is around 37%. Together these findings emphasize the value of user-level information for advertisers.

Next, we explore whether all advertisers benefit from a less restrictive data-sharing regime. In a competitive environment, greater ability to target does not necessarily translate into higher profits. Instead, it is the ability to target relative to competitors that matters. In Table 9, we show how many advertisers benefit as we move from one data-sharing scenario (column) to another (row).

	Full	Behavioral	Contextual	Baseline
Full	NA	28	36	35
Behavioral	9	NA	29	27
Contextual	1	8	NA	3
Baseline	2	10	19	NA

Table 9: Number of advertisers who benefit by moving from one data-sharing scenario (column) to another (row).

In general, more advertisers benefit when the platform shares more granular information. Moving from behavioral, contextual, and no data-sharing to full information sharing benefits 28, 36, and 35 advertisers respectively (first row of Table 9). This amounts to over 75% of the top 37 advertisers.

However, more information is not uniformly better for all advertisers. The lower triangle of Table 9 depicts situations where advertisers go from more data to less. Interestingly, it is populated with positive numbers, which suggests that individual advertisers often benefit from less information-sharing even as total advertiser surplus increases. For example, there are nine advertisers who prefer behavioral data-sharing to full data-sharing. Similarly, while the majority of advertisers prefer behavioral data-sharing, there is a small portion of advertisers who prefer contextual data-sharing. We present a simple example to highlight the intuition behind this – a nutrition supplement ad that advertises on a fitness app can get all the slots in that app at a low cost because other advertisers would place low bids when only app-level information is shared. However, this ad would be worse off if only behavioral information is shared, because the competition on the users of this app becomes more intense and this ad will no more be able to extract a large informational rent.

Overall, our findings offer some evidence that advertisers are likely to be differentially affected by privacy regulation on user-tracking and data-sharing. Indeed, some advertisers may prefer more regulation and privacy-preserving laws while others would prefer less. Understanding advertisers' incentives in this context and the drivers of heterogeneity in their preferences can thus help regulators craft the appropriate privacy policies in future.

7 Conclusions

Mobile in-app advertising is growing in popularity. In-app ads have unique tracking properties: they allow advertisers and ad-networks to access the device ID of the mobile devices showing the ads, and thereby enable high quality behavioral targeting. While this has made them appealing to advertisers, consumers privacy advocates are concerned about their invasiveness. Therefore, marketers and policy-makers are interested in understanding the relative effectiveness behavioral targeting compared to contextual targeting, which is privacy-preserving. These questions lead to a broader ongoing debate on the incentives of ad-networks to engage in behavioral targeting, to share user-level behavioral data with advertisers, and the role of regulation in preserving privacy.

We propose a modeling framework that consists of two components – a machine learning framework for click-through rate predictions and a stylized analytical framework for conducting data-sharing counterfactuals. We apply our framework to data from the leading in-app ad-network of an Asian country. We show that our machine learning model improves targeting ability by 17.95% over the baseline. These gains mainly stem from behavioral information and the value of contextual information is relatively small. Stricter regulations on user-tracking substantially shrink the value of behavioral targeting. Counterfactuals show that although total surplus grows with more granular information-sharing between the ad-network and advertisers, the ad-network's revenues are non-monotonic, i.e., it prefers to not share behavioral information. There is also some heterogeneity among advertisers' on their preferred level of data-sharing. Our findings suggest in the absence of data-sharing contracts between the ad-network and advertisers, the ad-network will naturally choose to preserve users' privacy.

Our paper makes three key contributions to the literature. From a substantive perspective, we quantify the relative value of different targeting information and present a comprehensive comparison of contextual vs. behavioral targeting. From a methodological standpoint, we leverage the randomization in our data to combine a predictive machine learning model with economic theory. Finally, from a policy point-of-view, we examine the incentives of two major parties, the platform and advertisers, on the optimal level of data-sharing. We expect our model and findings to speak to the debate on privacy and data-sharing regulations in mobile advertising marketplace.

Nevertheless, as discussed earlier, our paper suffers from limitations which serve as excellent avenues for future research. First, our analysis focuses only on clicks and does not capture subsequent actions. Future work could examine the effects of targeting and information disclosure on outcomes further down the purchase funnel (e.g., purchase/conversion). Second, in our counterfactuals, we assume that all advertisers are symmetric in their valuation for clicks. Relaxing this assumption and using a structural model to recover the distributions of valuations is a natural next step. Third, we investigate the case of symmetric information disclosure. However, the platform could asymmetrically share information with advertisers. Examining advertisers' strategic behavior in such environments could be an interesting exercise. Since information is a non-rival good, it also raises interesting questions on strategic information-sharing among advertisers. Pancras and Sudhir (2007) propose a framework to empirically study this problem for price-targeting. Future work could extend that work to the case of ad-targeting and compare it with price-targeting.

References

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

A. Ahmed, Y. Low, M. Aly, V. Josifovski, and A. J. Smola. Scalable Distributed Inference of Dynamic User Interests for Behavioral Targeting. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 114–122. ACM, 2011.

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

AppAnnie. In-App Advertising Spend to Triple, Reach $201 Billion by 2021, 2017. URL `https://www.appannie.com/en/insights/market-data/app-advertising-spend-2021/`.

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

R. Bekkerman. The Present and the Future of the KDD Cup Competition: an Outsider's Perspective, 2015. URL `https://www.linkedin.com/pulse/present-future-kdd-cup-competition-outsiders-ron-bekkerman`.

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

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

A. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A Semantic Approach to Contextual Advertising. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 559–566. ACM, 2007.

R. Caruana and A. Niculescu-Mizil. An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 161–168. ACM, 2006.

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

D. Chaffey. Mobile Marketing Statistics Compilation, 2017. URL `http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/`.

D. Chakrabarti, D. Agarwal, and V. Josifovski. Contextual Advertising by Combining Relevance with Click Feedback. In *Proceedings of the 17th international conference on World Wide Web*, pages 417–426. ACM, 2008.

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

T. Chen and C. Guestrin. Xgboost: A Scalable Tree Boosting System. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

Y. Chen, C. Narasimhan, and Z. J. Zhang. Individual Marketing with Imperfect Targetability. *Marketing Science*, 20(1):23–41, 2001.

Y. Chen, M. Kapralov, J. Canny, and D. Y. Pavlov. Factor Modeling for Advertisement Targeting. In *Advances in Neural Information Processing Systems*, pages 324–332, 2009.

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

D. Dzyabura and J. R. Hauser. Active Machine Learning for Consideration Heuristics. *Marketing Science*, 30 (5):801–819, 2011.

J. Edwards. Apple Has Quietly Started Tracking iPhone Users Again, And It's

Tricky To Opt Out, 2012. URL http://www.businessinsider.com/ifa-apples-iphone-tracking-in-ios-6-2012-10.

A. Edwards-Levy and D. Liebelson. Even Trump Voters Hate This Bill He Just Signed, 2017. URL http://www.huffingtonpost.com/entry/trump-online-privacy-poll_us_58e295e7e4b0f4a923b0d94a.

eMarketer. Worldwide Internet and Mobile Users: eMarketer's Estimates for 2016–2021 , 2017a. URL http://totalaccess.emarketer.com/reports/viewer.aspx?r=2002038.

eMarketer. Worldwide Ad Spending: eMarketer's Updated Estimates and Forecast for 2016–2021 , 2017b. URL http://totalaccess.emarketer.com/reports/viewer.aspx?r=2002145.

T. Evgeniou, M. Pontil, and O. Toubia. A convex optimization approach to modeling consumer heterogeneity in conjoint estimation. *Marketing Science*, 26(6):805–818, 2007.

A. Farahat and M. C. Bailey. How effective is targeted advertising? In *Proceedings of the 21st International Conference on World Wide Web*, pages 111–120. ACM, 2012.

Y. Freund. Boosting a Weak Learning Algorithm by Majority. *Information and Computation*, 121(2): 256–285, 1995.

Y. Freund and R. E. Schapire. Experiments with a New Boosting Algorithm. In *ICML*, volume 96, pages 148–156, 1996.

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

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

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

B. Fung. What to Expect Now That Internet Providers Can Collect and Sell Your Web Browser History, 2017. URL https://www.washingtonpost.com/news/the-switch/wp/2017/03/29/what-to-expect-now-that-internet-providers-can-collect-and-sell-your-web-browser ?tid=hybrid_collaborative_1_na&utm_term=.8cae2f7af071.

J.-J. Ganuza. Ignorance Promotes Competition: An Auction Model with Endogenous Private Valuations. *Rand Journal of Economics*, pages 583–598, 2004.

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

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

A. Goldfarb and C. E. Tucker. Privacy Regulation and Online Advertising. *Management science*, 57(1): 57–71, 2011b.

S. Gray. iOS 10 to Feature Stronger "Limit Ad Tracking" Control, 2016. URL https://fpf.org/2016/08/02/ios-10-feature-stronger-limit-ad-tracking/.

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

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

R. Hollander. Apps Are Becoming Extremely Reliant on In-

App Ads, 2017. URL http://www.businessinsider.com/apps-are-becoming-extremely-reliant-on-in-app-ads-2017-8.

D. Huang and L. Luo. Consumer Preference Elicitation of Complex Products Using Fuzzy Support Vector Machine Active Learning. *Marketing Science*, 35(3):445–464, 2016.

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

G. Iyer, D. Soberman, and J. M. Villas-Boas. The Targeting of Advertising. *Marketing Science*, 24(3): 461–476, 2005.

G. A. Johnson. The Impact of Privacy Policy on the Auction Market for Online Display Advertising. 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. Kint. Opinion: Europe's Strict New Privacy Rules Are Scary but Right, 2017. URL http://adage.com/article/digitalnext/europe-s-strict-privacy-rules-terrifying-apple/309155/.

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.

L. Liu and D. Dzyabura. Capturing Multi-taste Preferences: A Machine Learning Approach. Working Paper, 2016.

S. Lu and S. Yang. A Two-Sided Market Analysis of Behaviorally Targeted Display Advertising. 2015.

V. Marotta, K. Zhang, and A. Acquisti. Not All Privacy Is Created Equal: The Welfare Impact of Targeted Advertising. 2017.

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

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

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

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

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

J. Pancras and K. Sudhir. Optimal Marketing Strategies for a Customer Data Intermediary. *Journal of Marketing research*, 44(4):560–578, 2007.

S. Perez. Consumers Spend 85% Of Time On Smartphones In Apps, But Only 5 Apps See Heavy Use, 2015. URL https://techcrunch.com/2015/06/22/consumers-spend-85-of-time-on-smartphones-in-apps-but-only-5-apps-see-heavy-use/

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

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

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

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

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

R. E. Schapire. The Strength of Weak Learnability. *Machine Learning*, 5(2):197–227, 1990.

E. B. Seufert. IDFA Zeroing is the Massive Change to Mobile Advertising That No One is Talking About, 2016. URL `http://mobiledevmemo.com/idfa-zeroing-massive-change/`.

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

L. Stampler. Here's Everything We Know About IFA, The iPhone Tracking Technology In Apple's iOS 6, 2012. URL `http://www.businessinsider.com/everything-we-know-about-ifa-and-tracking-in-apples-ios-6-2012-10`.

G. Sterling. Google Replacing "Android ID" With "Advertising ID" Similar To Apple's IDFA, 2013. URL `http://marketingland.com/google-replacing-android-id-with-advertising-id-similar-to-apples-idfa-63636`.

O. Toubia, D. I. Simester, J. R. Hauser, and E. Dahan. Fast Polyhedral Adaptive Conjoint Estimation. *Marketing Science*, 22(3):273–303, 2003.

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

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

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

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

H. Yoganarasimhan. Search Personalization Using Machine Learning. *Available at SSRN 2590020*, 2017.

Appendices

A Validation

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

As discussed in §4.3, 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.

B Table of Features

Table A1: List of Features.

No.	Feature Name	Feature Classification			Contextual Features	
		Behavioral	Contextual	Ad-specific	App-specific	Time-specific
1	$Impressions(\varnothing, \varnothing, \varnothing, u; l, r)$	✓				
2	$Impressions(a, \varnothing, \varnothing, \varnothing; l, g)$			✓		
3	$Impressions(\varnothing, p, \varnothing, \varnothing; l, g)$		✓		✓	
4	$Impressions(\varnothing, \varnothing, t, \varnothing; l, g)$		✓			✓
5	$Impressions(a, \varnothing, \varnothing, u; l, r)$	✓		✓		
6	$Impressions(\varnothing, p, \varnothing, u; l, r)$	✓	✓		✓	
7	$Impressions(\varnothing, \varnothing, t, u; l, r)$	✓	✓			✓
8	$Impressions(a, p, \varnothing, \varnothing; l, g)$		✓	✓	✓	
9	$Impressions(a, \varnothing, t, \varnothing; l, g)$		✓	✓		✓
10	$Impressions(\varnothing, p, t, \varnothing; l, g)$		✓		✓	✓
11	$Impressions(a, p, \varnothing, u; l, r)$	✓	✓	✓	✓	
12	$Impressions(a, p, t, \varnothing; l, g)$		✓	✓	✓	✓
13	$Impressions(\varnothing, \varnothing, \varnothing, u; s, r)$	✓				
14	$Impressions(a, \varnothing, \varnothing, \varnothing; s, g)$			✓		
15	$Impressions(\varnothing, p, \varnothing, \varnothing; s, g)$		✓		✓	
16	$Impressions(\varnothing, \varnothing, t, \varnothing; s, g)$		✓			✓
17	$Impressions(a, \varnothing, \varnothing, u; s, r)$	✓		✓		
18	$Impressions(\varnothing, p, \varnothing, u; s, r)$	✓	✓		✓	
19	$Impressions(\varnothing, \varnothing, t, u; s, r)$	✓	✓			✓
20	$Impressions(a, p, \varnothing, \varnothing; s, g)$		✓	✓	✓	
21	$Impressions(a, \varnothing, t, \varnothing; s, g)$		✓	✓		✓
22	$Impressions(\varnothing, p, t, \varnothing; s, g)$		✓		✓	✓
23	$Impressions(a, p, \varnothing, u; s, r)$	✓	✓	✓	✓	
24	$Impressions(a, p, t, \varnothing; s, g)$		✓	✓	✓	✓
25	$Impressions(\varnothing, \varnothing, \varnothing, u; o, r)$	✓				
26	$Impressions(a, \varnothing, \varnothing, u; o, r)$	✓		✓		
27	$Clicks(\varnothing, \varnothing, \varnothing, u; l, r)$	✓				
28	$Clicks(a, \varnothing, \varnothing, \varnothing; l, g)$			✓		
29	$Clicks(\varnothing, p, \varnothing, \varnothing; l, g)$		✓		✓	
30	$Clicks(\varnothing, \varnothing, t, \varnothing; l, g)$		✓			✓
31	$Clicks(a, \varnothing, \varnothing, u; l, r)$	✓		✓		
32	$Clicks(\varnothing, p, \varnothing, u; l, r)$	✓	✓		✓	
33	$Clicks(\varnothing, \varnothing, t, u; l, r)$	✓	✓			✓
34	$Clicks(a, p, \varnothing, \varnothing; l, g)$		✓	✓	✓	
35	$Clicks(a, \varnothing, t, \varnothing; l, g)$		✓	✓		✓
36	$Clicks(\varnothing, p, t, \varnothing; l, g)$		✓		✓	✓
37	$Clicks(a, p, \varnothing, u; l, r)$	✓	✓	✓	✓	
38	$Clicks(a, p, t, \varnothing; l, g)$		✓	✓	✓	✓
39	$Clicks(\varnothing, \varnothing, \varnothing, u; s, r)$	✓				
40	$Clicks(a, \varnothing, \varnothing, \varnothing; s, g)$			✓		
41	$Clicks(\varnothing, p, \varnothing, \varnothing; s, g)$		✓		✓	
42	$Clicks(\varnothing, \varnothing, t, \varnothing; s, g)$		✓			✓
43	$Clicks(a, \varnothing, \varnothing, u; s, r)$	✓		✓		

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
44	$Clicks(\varnothing, p, \varnothing, u; s, r)$	✓	✓		✓	
45	$Clicks(\varnothing, \varnothing, t, u; s, r)$	✓	✓			✓
46	$Clicks(a, p, \varnothing, \varnothing; s, g)$		✓	✓	✓	
47	$Clicks(a, \varnothing, t, \varnothing; s, g)$		✓	✓		✓
48	$Clicks(\varnothing, p, t, \varnothing; s, g)$		✓		✓	✓
49	$Clicks(a, p, \varnothing, u; s, r)$	✓	✓	✓	✓	
50	$Clicks(a, p, t, \varnothing; s, g)$		✓	✓	✓	✓
51	$CTR(\varnothing, \varnothing, \varnothing, u; l, r)$	✓				
52	$CTR(a, \varnothing, \varnothing, \varnothing; l, g)$			✓		
53	$CTR(\varnothing, p, \varnothing, \varnothing; l, g)$		✓		✓	
54	$CTR(\varnothing, \varnothing, t, \varnothing; l, g)$		✓			✓
55	$CTR(a, \varnothing, \varnothing, u; l, r)$	✓		✓		
56	$CTR(\varnothing, p, \varnothing, u; l, r)$	✓	✓		✓	
57	$CTR(\varnothing, \varnothing, t, u; l, r)$	✓	✓			✓
58	$CTR(a, p, \varnothing, \varnothing; l, g)$		✓	✓	✓	
59	$CTR(a, \varnothing, t, \varnothing; l, g)$		✓	✓		✓
60	$CTR(\varnothing, p, t, \varnothing; l, g)$		✓		✓	✓
61	$CTR(a, p, \varnothing, u; l, r)$	✓	✓	✓	✓	
62	$CTR(a, p, t, \varnothing; l, g)$		✓	✓	✓	✓
63	$CTR(\varnothing, \varnothing, \varnothing, u; s, r)$	✓				
64	$CTR(a, \varnothing, \varnothing, \varnothing; s, g)$			✓		
65	$CTR(\varnothing, p, \varnothing, \varnothing; s, g)$		✓		✓	
66	$CTR(\varnothing, \varnothing, t, \varnothing; s, g)$		✓			✓
67	$CTR(a, \varnothing, \varnothing, u; s, r)$	✓		✓		
68	$CTR(\varnothing, p, \varnothing, u; s, r)$	✓	✓		✓	
69	$CTR(\varnothing, \varnothing, t, u; s, r)$	✓	✓			✓
70	$CTR(a, p, \varnothing, \varnothing; s, g)$		✓	✓	✓	
71	$CTR(a, \varnothing, t, \varnothing; s, g)$		✓	✓		✓
72	$CTR(\varnothing, p, t, \varnothing; s, g)$		✓		✓	✓
73	$CTR(a, p, \varnothing, u; s, r)$	✓	✓	✓	✓	
74	$CTR(a, p, t, \varnothing; s, g)$		✓	✓	✓	✓
75	$AdCount(\varnothing, u; l, g)$	✓		✓		
76	$AdCount(p, \varnothing; l, g)$		✓	✓	✓	
77	$AdCount(p, u; l, g)$	✓	✓	✓	✓	
78	$AdCount(\varnothing, u; s, g)$	✓		✓		
79	$AdCount(p, \varnothing; s, g)$		✓	✓	✓	
80	$AdCount(p, u; s, g)$	✓	✓	✓	✓	
81	$AdCount(\varnothing, u; o, r)$	✓		✓		
82	$AppCount(\varnothing, u; l, g)$	✓	✓		✓	
83	$AppCount(a, \varnothing; l, g)$		✓	✓	✓	
84	$AppCount(a, u; l, g)$	✓	✓	✓	✓	
85	$AppCount(\varnothing, u; s, g)$	✓	✓		✓	
86	$AppCount(a, \varnothing; s, g)$		✓	✓	✓	
87	$AppCount(a, u; s, g)$	✓	✓	✓	✓	
88	$Entropy(\varnothing, u; l, g)$	✓		✓		
89	$Entropy(p, \varnothing; l, g)$		✓	✓	✓	
90	$Entropy(p, u; l, g)$	✓	✓	✓	✓	

Continued on next page

Table A1 – continued from previous page

No.	Feature Name	Feature Classification			Contextual Features	
		Behavioral	Contextual	Ad-specific	App-specific	Time-specific
91	$Entropy(\varnothing, u; s, g)$	✓		✓		
92	$Entropy(p, \varnothing; s, g)$		✓	✓	✓	
93	$Entropy(p, u; s, g)$	✓	✓	✓	✓	
94	$Entropy(\varnothing, u; o, r)$	✓		✓		
95	$TimeVariability(u; l, g)$	✓	✓			✓
96	$TimeVariability(u; s, g)$	✓	✓			✓
97	$AppVariability(u; l, g)$	✓	✓		✓	
98	$AppVariability(u; s, g)$	✓	✓		✓	
99	$Latitude(u)$	✓				
100	$Longitude(u)$	✓				
101	$WiFi(u)$	✓				
102-109	$Brand(u)$	✓				
110-113	$Operator(u)$	✓				
114-122	$ISP(u)$	✓				
123	$Bid(a)$			✓		
124-161	$AdDummy(a)$			✓		

C Appendix for Robustness Checks

C.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 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

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
Full Sample	0/1 Improvement (%)	0.63%	0.00%	4.74%
Top Ads/Apps	0/1 Improvement (%)	1.07%	0.00%	8.23%
Full Sample	MSE Improvement (%)	3.41%	0.55%	8.59%
Top Ads/Apps	MSE Improvement (%)	4.86%	0.67%	13.33%
Full Sample	RIG	12.14%	5.25%	17.95%
Top Ads/Apps	RIG	14.82%	5.98%	22.85%

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

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).

We now take the optimized models presented in §5 and evaluate their performance on these alternative metrics. Table A2 is analogous to Table 4 in the main text and Table A3 and A4 are analogous to Table 5. By and large, all our substantive results remain the same when we use the alternative evaluation metrics.

Model	**Feature Set**	**AUC**	**0/1 Loss**		**Mean-Squared Error**	
			Value	% Improvement	Value	% Improvement
Ad	$F_A \setminus (F_P \cup F_T \cup F_B)$	0.5918	0.009582	0.00%	0.009479	0.11%
App	$F_P \setminus (F_A \cup F_T \cup F_B)$	0.6745	0.009582	0.00%	0.009446	0.46%
Time	$F_T \setminus (F_A \cup F_P \cup F_B)$	0.5336	0.009582	0.00%	0.009489	0.01%
User	$F_B \setminus (F_A \cup F_P \cup F_T)$	0.7751	0.009568	0.14%	0.009269	2.32%

Table A3: Performance of Forward Addition Model with different pieces of information with on other metrics

Model	**Feature Set**	**AUC**	**0/1 Loss**		**Mean-Squared Error**	
			Value	% Improvement	Value	% Improvement
Ad	$F_F \setminus F_A$	0.8196	0.009239	3.57%	0.008753	7.76%
App	$F_F \setminus F_P$	0.8000	0.009249	3.47%	0.008842	6.82%
Time	$F_F \setminus F_T$	0.8191	0.009422	1.67%	0.008981	5.37%
User	$F_F \setminus F_B$	0.7042	0.009582	0.00%	0.009437	0.55%

Table A4: Performance of Backward Deduction Model with different pieces of information with on other metrics

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 A5: *RIG* of different learning methods for the test data.

C.2 Other Learning Methods

We now compare the performance of XGBoost with other five other learning algorithms and present the results in A5. Note that XGBoost outperforms all of them.

Because each learning model has different set of hyper-parameters, we now describe the hyper-parameters associated with each model and our approach to choosing the optimal hyper-parameters in each case. Note that in all the cases, we use the same high-level validation procedure described in §A.

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

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 A6: 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.

C.3 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.[24] 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 A6. 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 A6 suggests that after about 100,000 users, increasing the sample size improves the prediction only slightly. However, increasing sample sizes also increase the training time and computational costs. Given the cost-benefit trade-off, our sample of 727,354 users is more than sufficient for our purposes.

C.4 Other Validation Techniques

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

[24] 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.

Identifier	**RIG over Baseline**	**No. of Features used in Training**
No Identifier	5.29%	87
IP Address	12.64%	161
AAID	17.18%	161
Both Identifiers	17.24%	259

Table A7: RIG of models with different identifiers

- 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 (2017) and Hastie et al. (2001) for a detailed discussion on the pros and cons of k-fold cross validation.

C.5 Comparison of User Identifiers on Same Data

To ensure that the differences in the results from using IP as a user-identifier compared to using AAID as a user-identifier (i.e., the differences in Tables 4 and 6) are not driven by the differences in samples, we conduct some additional checks. Recall that we had sampled set of 728,340 unique AAIDs for the main analysis and another set of 799,219 unique IPs for the analysis above. The results of Table 4 are based on the former and the results in Table 6 are based on the latter. Around 15% of the impressions in these two sets are mutual, i.e., included in both samples. This overlapping set contains 4,178,828 impressions, of which 2,713,823 of are in training and validation data, and 1,465,005 in the test data.

We now train four different models, each with features generated using different identifiers and examine the performance of each on the test data from this overlapping data-set. The results from this exercise are presented in Table A7. The third column refers to the number of features which are used to train the model. First, we find that the use of any identifier results in a large information gain; all the models with identifiers perform better than the first model with no identifier. Second, we find that the AAID helps more than IP: the model with AAID as the identifier has an *RIG* of 17.18%, whereas this number drops to 12.64% when we use IP as the main identifier. This reaffirms the importance of using AAID for user-identification in mobile advertising and suggests that IP cannot function as a reasonable substitute. We also consider the model with both identifiers to see if the combination performs better. In this case, we include two sets of behavioral features – one set generated using AAID as the user-identifier and another set generated using IP as the user-identifier. Since user-level features are duplicated, the total number of features expands to 259 in this case. Here, we find very small improvement over simply using AAID-based behavioral features, implying that IP does not add much information over and above AAID.

	Ad 1	Ad 2	Ad 3	Ad 4
Full			0.012	
Behavioral			0.011	
Contextual			0.014	
Baseline			0.018	

Step 1: Predictions of the actual data

	Ad 1	Ad 2	Ad 3	Ad 4
Full	0.034	0.025	0.012	0.005
Behavioral	0.028	0.031	0.011	0.004
Contextual	0.017	0.018	0.014	0.006
Baseline	0.010	0.012	0.018	0.005

Step 2: Counterfactual predictions

	Ad 1	Ad 2	Ad 3	Ad 4
Full	1	0	1	0
Behavioral	1	0	1	0
Contextual	1	0	1	0
Baseline	1	0	1	0

Step 3: Filtering

	Ad 1	Ad 2	Ad 3	Ad 4
Full	0.034	0	0.012	0
Behavioral	0.028	0	0.011	0
Contextual	0.017	0	0.014	0
Baseline	0.010	0	0.018	0

Step 4: Final match values

	Winning Ad	Platform Revenues	Total Surplus	Advertiser Surplus
Full	1	0.012	0.034	0.022
Behavioral	1	0.011	0.034	0.023
Contextual	1	0.014	0.034	0.020
Baseline	3	0.010	0.012	0.002

Step 5: Analysis of the competition

Figure 8: Example showing step-by-step counterfactual procedure for one impression and four ads.

D Counterfactual Predictions

D.1 Overview

We now discuss the procedure for generating counterfactual estimates of platform revenues, advertisers' revenues, and total surplus under four data-sharing scenarios – no data-sharing (baseline), contextual data-sharing, behavioral data-sharing, and full data-sharing. There is a one-to-one correspondence between these data-sharing scenarios and the targeting models from §5. The no data-sharing scenario corresponds to the baseline or no targeting model, contextual data-sharing to the Contextual targeting model, behavioral data-sharing to the Behavioral targeting model, and full data-sharing to the Full targeting model.

There are three main steps in our counterfactual procedure – 1) generating the match value matrices for different data-sharing scenarios, 2) filtering, and 3) revenue and surplus estimation. To give an overview of our counterfactual procedure, we start with a simple example of an impression with four potential ads. Figure 8 sketches how we generate counterfactuals and analyze the competition for this particular impression. Suppose that Ad 3 has been shown in this impression in the actual data. Therefore, using our predictions obtained from different models in §5, we can estimate the match values for Ad 3 under all scenarios as shown in Step 1 in Figure 8. Next, we need to estimate match values for counterfactual ads, *i.e.*, those that were not shown in this impression in the data. To do so, we need to re-generate all the ad-specific features, while keeping all the features with $a = \varnothing$ unchanged. After re-generating ad-specific features, we use the targeting from §5 to estimate the match values for these counterfactual ads, as shown in Step 2 in Figure 8.

However, the estimates of match values or eCTR of some counterfactual ads based on our targeting models can be inaccurate. As discussed in §3.5, the predictive accuracy of our models is predicated on the joint distribution of co-variates and clicks in the test data being the same as that in training data. It implies that the estimates of match value in Step 2 might not be accurate if those counterfactual impressions are not drawn from the joint distribution of co-variates and clicks in the training set. To satisfy this condition, we use a filtering procedure which screens out the ads that would not have been shown in the focal impression due to their targeting decisions. In our example, let us suppose that Ad 2 and Ad 4 do not bid on this particular impression. As a result, this impression with either Ad 2 or Ad 4 being shown is not drawn from the joint distribution on which we have trained our models. Therefore, we filter out these two ads as shown in Step 3 and 4 in Figure 8. The idea is to only consider ads that would have been shown in that impression, *i.e.*, those that are actually competing for that impression. Given the probabilistic nature of auction used by the platform, we can then satisfy the joint distribution condition and guarantee correct match value estimates.

Finally, using the match value estimates for all unfiltered ads, we determine the winning ad, platform revenues, total surplus, and advertiser's surplus. Note that we have assumed a second price auction wherein advertisers are symmetric in their valuation of click. So in each data-sharing scenario, the winning ad is the one with the highest match value in that scenario and it pays the platform the second highest match value. The total surplus is the match value of the winning ad under full data-sharing because it is our best estimate of the actual likelihood of a click. Finally, the winning advertiser's surplus can be calculated by subtracting the total surplus and the platform's revenue. This analysis for our example is shown in Step 5 of Figure 8.

D.2 Step by Step Procedure for Generating Counterfactuals

Let $\tilde{M}^{\mathcal{F}}$, $\tilde{M}^{\mathcal{B}}$, $\tilde{M}^{\mathcal{C}}$, and $\tilde{M}^{\mathcal{N}}$ denote the counterfactual match value matrices for full, behavioral, contextual, and no (baseline) data-sharing respectively. These matrices each have 7,475,908 rows (number of impressions in the test set with top apps, i.e., $p_i \neq p^{(s)}$ for impression i) and 37 columns (top ads, i.e., $a_i \neq a^{(s)}$ for impression i). We further define E as our filtering matrix. Finally, let $\hat{M}^{\mathcal{F}}$ (full), $\hat{M}^{\mathcal{B}}$ (behavioral), $\hat{M}^{\mathcal{C}}$ (contextual), and $\hat{M}^{\mathcal{N}}$ (baseline) denote our final estimated match value matrices. We now present a step by step procedure for generating each element in the matrices defined above:

1. Generating counterfactual match value matrices: for each row (impression) i and column (ad) j in the match value matrices:
   - Generate the features for impression i as though ad j is being shown in this impression. It is worth noting that only ad-specific features will change. Other features with $a = \varnothing$ remain the same as data.
   - Use the Contextual, Behavioral, and Full models to estimate the match or eCTR for impression i and ad j $\forall\, i, j$. Note that this is possible because all the features for a given impression-ad pair are available. Let $\tilde{m}^{\mathcal{C}}_{ij}$, $\tilde{m}^{\mathcal{B}}_{ij}$, and $\tilde{m}^{\mathcal{F}}_{ij}$ denote the estimated match value matrices for the Contextual, Behavioral, and Full models. Next, to estimate the match value matrix for the no targeting case, set the elements of $\tilde{m}^{\mathcal{N}}_{ij}$ as the average past CTR of ad j for all impressions. This is our baseline prediction, under the no data-sharing scenario.
2. Filtering counterfactual impressions: for each row i and column j in the filtering matrix:

- Filter the ads that would have not been shown in impression i using the filtering matrix E. For example, if ad j is excluded from the competition for impression i due to its targeting decisions, $e_{ij} = 0$. Else, $e_{ij} = 1$.
- Obtain final counterfactual values $\hat{m}_{ij}^{\mathcal{T}} = \tilde{m}_{ij}^{\mathcal{T}} e_{ij}$, where $\mathcal{T} \in \{\mathcal{N}, \mathcal{C}, \mathcal{B}, \mathcal{F}\}$.

3. Estimating revenues and surplus: after obtaining counterfactual matrices, we analyze the competition for each impression i. Let $a_i^{\mathcal{T}}$, $R_i^{\mathcal{T}}$, and $S_i^{\mathcal{T}}$ denote the the winning ad, platform's revenue, and total surplus under scenario $\mathcal{T}$ respectively. We also define $W_{ij}^{\mathcal{T}}$ as advertiser j's surplus from impression i. For each impressions i, we calculate these metrics as follows:

   - Winning ad: the winning ad under scenario $\mathcal{T}$ is the ad with the highest match value for i, *i.e.*, $a_i^{\mathcal{T}} = \text{argmax}_j \, \hat{m}_{ij}^{\mathcal{T}}$. Note that the winning ad is determined using a second-price auction. Therefore, there is no uncertainty in who will win the impression in the counterfactuals.
   - Platform's revenue: since the platform runs a second-price auction, the winning ad will pay the second highest bid. Formally, the revenue that the platform collects for impression i under scenario $\mathcal{T}$ is:
$$R_i^{\mathcal{T}} = \max_{j \backslash a_i^{\mathcal{T}}} \hat{m}_{ij}^{\mathcal{T}} \tag{25}$$
   - Total surplus: it is the total value created by impression i. Hence, it is the match value of the winning ad which is shared between the platform and advertisers. We can write:
$$S_i^{\mathcal{T}} = \hat{m}_{ia_i^{\mathcal{T}}}^{\mathcal{F}} \tag{26}$$
     It is worth mentioning that it is the match value of winning ad estimated by the full model, since it is the most accurate model we have.
   - Advertiser's surplus: in each scenario, if an advertiser wins an impressions, its surplus is the difference between the total surplus and platform's revenue for that impression. If the advertiser does not win the impression, its surplus for the impression is zero. For an advertiser j, we can write her surplus for impression i as follows:
$$W_{ij}^{\mathcal{T}} = (S_i^{\mathcal{T}} - R_i^{\mathcal{T}}) \mathbb{1}\big(a_i^{\mathcal{T}} = a^{(j)}\big) \tag{27}$$

   Using Eq. 25, 26, and 27, we can derive the total platform's revenue, total surplus, and advertiser j's under scenario $\mathcal{T}$ (denoted as $R^{\mathcal{T}}$, $S^{\mathcal{T}}$, and $W_j^{\mathcal{T}}$ respectively) as follows:
$$R^{\mathcal{T}} = \sum_{i=1}^{I} R_i^{\mathcal{T}} = \sum_{i=1}^{I} \max_{j \backslash a_i^{\mathcal{T}}} \hat{m}_{ij}^{\mathcal{T}}$$
$$S^{\mathcal{T}} = \sum_{i=1}^{I} S_i^{\mathcal{T}} = \sum_{i=1}^{I} \hat{m}_{ia_i^{\mathcal{T}}}^{\mathcal{F}}$$
$$W_j^{\mathcal{T}} = \sum_{i=1}^{I} W_{ij}^{\mathcal{T}} = \sum_{i=1}^{I} (S_i^{\mathcal{T}} - R_i^{\mathcal{T}}) \mathbb{1}\big(a_i^{\mathcal{T}} = a^{(j)}\big) \ , \ \forall \, j \in \{1, \dots, 37\}$$

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

Table A8: Example depicting two scenarios under CPC mechanism: 1) No targeting and 2) Perfect Targeting

E 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_{ij} and m_{ij} respectively denote the valuation for click and the match value for advertiser j for impression i. The maximum revenue that platform could get is then $\max_j v_{ij} m_{ij}$. Thus, defining b_{ij} as advertiser j's bid on impression i, the platform should sell the ad slot to $\operatorname{argmax}_j b_{ij} m_{ij}$ and charge her the minimum bid with which she still wins. It generates the expected revenue of the second-highest $b_{ij} m_{ij}$. (In fact, this is how Google's sponsored search auctions work.)

Table A8 shows how the example in §6.1 generalizes to the CPC case. Although CPC and CPI have different properties, we can easily prove that their revenue is the same under different levels of targeting. This stems from the fact that under second-price auction, bidders bid their valuation. Thus, the platform's expected revenue from impression i under both mechanisms is the second highest $v_{ij}m_{ij}$, 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 data-sharing strategy and resulting match-value matrix. In the CPI mechanism, m_{ij} 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 withhold behavioral targeting information from advertisers. 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.

References

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

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.

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.

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. *Available at SSRN 2590020*, 2017.

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