A Practical Framework of Conversion Rate Prediction for Online Display Advertising

Quan Lu, Shengjun Pan, Liang Wang, Junwei Pan, Fengdan Wan
Yahoo! Inc.
701 First Ave
Sunnyvale, California, USA 94089
{qlu,alanpan,wlliang,jwpan,fengdanwan}@yahoo-inc.com

Hongxia Yang
Alibaba Group
969 West Wen Yi Road
Hangzhou, Zhejiang, China 311121
yang.yhx@alibaba-inc.com

ABSTRACT

Cost-per-action (CPA), or cost-per-acquisition, has become the primary campaign performance objective in online advertising industry. As a result, accurate conversion rate (CVR) prediction is crucial for any real-time bidding (RTB) platform. However, CVR prediction is quite challenging due to several factors, including extremely sparse conversions, delayed feedback, attribution gaps between the platform and the third party, etc. In order to tackle these challenges, we proposed a practical framework that has been successfully deployed on *Yahoo! BrightRoll*, one of the largest RTB ad buying platforms. In this paper, we first show that over-prediction and the resulted over-bidding are fundamental challenges for CPA campaigns in a real RTB environment. We then propose a safe prediction framework with conversion attribution adjustment to handle over-predictions and to further alleviate over-bidding at different levels. At last, we illustrate both offline and online experimental results to demonstrate the effectiveness of the framework.

CCS CONCEPTS

•Information systems →Computational advertising; Display advertising;

KEYWORDS

display advertising, demand-side platform, real-time bidding, large-scale learning

ACM Reference format:
Quan Lu, Shengjun Pan, Liang Wang, Junwei Pan, Fengdan Wan and Hongxia Yang. 2017. A Practical Framework of Conversion Rate Prediction for Online Display Advertising. In *Proceedings of The 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining', Halifax, Nova Scotia - Canada, July 2017 (KDD),* 9 pages.
DOI: 10.475/123_4

1 INTRODUCTION

In the past few years, advertisers have been rapidly shifting their media buying budgets to programmatic ad buying via RTB protocol. With different product goals in mind, advertisers can start campaigns with different goal types, including cost-per-milli (CPM) model, which are priced in bundles of 1,000 impressions (or ads delivery) and cost-per-click (CPC) or cost-per-action (CPA), which are priced by the resulted clicks or conversions. Among these goal types, CPA campaigns have become dominant due to its direct effects on advertisers' true return on investment (ROI). Especially they are also less affected by notorious online frauds.

Partial of the work was done when the last author was working at Yahoo! Inc.

KDD, Halifax, Nova Scotia - Canada

DOI: 10.475/123_4

1.1 CPA Campaign Setup

To run a CPA campaign in a buying platform, the first step is to set up a pixel to track conversion events. This is typically done by inserting a piece of JavaScript code at a specific page in an advertiser's website. After the pixel installation, any load of this page triggers the embedded JavaScript code, which fetches user's cookie ID, along with some other information, and sends it back to the platform. This is counted as one *global conversion* since it has yet to be credited to a winning platform by a third-party company. On the other side, the platform needs to figure out which global conversions may be attributed to itself periodically. Depending on the preset attribution rules, the attribution process traces back from the conversion time with a predefined time window to locate ads or clicks generated by the platform. If any shown ad or click happened within the time window, this global conversion is considered as a *local attribution* by the platform. However, whether this conversion will be attributed to the platform can only be determined by the third-party company who has a panorama view of ad events from all buying platforms. For any single buying platform, there is no way to be ascertain that a global conversion will finally be attributed to it. The local attribution process is invoked after a user visits a page that triggers the firing of the conversion pixel, and followed by all subsequent updates happening in CVR prediction process. For CPA campaigns, a widely-used performance metric is the effective cost per action or acquisition (eCPA) [1]

$$eCPA = \frac{\text{total cost of showing ads}}{\text{total number of actions}},$$

which tells the advertisers the actual inventory acquisition cost they spend for each action.

1.2 Challenges

On the surface, CVR prediction may look very similar to the well studied CTR prediction problem. As a consequence, there is far less literature regarding CVR prediction and most researches consider CVR prediction as a natural extension from CTR prediction. However, a conversion requires more user engagement than a click in terms of time and/or monetary spending. Such a costly event requires a user to demonstrate stronger intention signals in advance and thus users' behavior-related features are more important. As the industry leading programmatic ad buying platform, *Yahoo! Brightroll* automates RTB buying processes for thousands of CPA campaigns and provides predictions on billions of events everyday. We argue that when developing a CVR prediction model used in the real RTB production environment, the following challenges must be tackled:

Conversion Rarity Compared with CTR, CVR is usually several magnitude smaller. This kind of extreme rarity makes CVR prediction much more challenging. As elaborated later in Section 3, over-predictions are more commonly observed in CVR prediction and consequently depart campaigns actual eCPAs far away from the advertisers' goals.

[1] http://cpm.wiki/define/eCPA

Delayed Conversion Feedback The time length between an impression and its click is usually seconds, but the time length between an impression and the resulted conversion could be hours or even days. This difference leads to two complications. First, the conversion model needs to be built to predict events happening in a much later future time. Second, algorithms with non-delayed performance feedback assumptions are very unlikely to be applicable anymore.

Local vs. Global Contributions Another challenge that impacts CVR estimation is *local attribution*. As explained before, we are only certain of *local attribution*, which usually results in an inflated number of attributed conversions compared to the final attributed conversions from the third party company. This discrepancy sometimes can be unexpectedly large.

Impression Value Divergence In display ad systems, the inventory cost generally follows the second-price auction model and the best practice is to bid with the true value [13]. For a CPC goal ad, the estimated true value is fully determined by the predicted CTR. However, due to the existence of attribution, the true value estimation for a CPA campaign ad is not only dependent on the predicted CVR but also other factors like the "residual value" from the previous shown ads.

The rest of this paper is organized as follows. In Section 2, we review related work. The CVR over-prediction issue is explored in Section 3. We argue that over-prediction is extremely detrimental for CPA campaigns' performance and any well-designed CVR prediction model must handle it properly. In Section 4, we propose a practical framework that can tackle these practical challenges which has been successfully deployed on *Yahoo! BrightRoll*. Section 5 shows how to further reduce prediction bias and adjust the bidding price caused by attribution allocations. Experimental results and evaluations are presented in Section 6. Finally, Section 7 concludes the paper.

2 RELATED WORK

There has been extensive research [7, 11, 14] on CTR prediction for both search and display advertising. Since both clicks and conversions are post-impression events, it's a natural idea to borrow the well-studied modeling techniques used for CTR predictions and apply them directly to CVR predictions [16]. However, as summarized in Section 1.2, considerations have been taken into to address those challenges in order to successfully achieve campaign goals. Very few studies focus on these challenges and the following is a complete list to the best of our knowledge. [10] studied the delay between a conversion and previous impressions with the focus on designing mechanisms for CPA campaigns but not CPA predictions. [8] discussed the difference in incentives between CPC and CPA from economic point of view. [4] focused on post-click CPA and compared the differences in click-to-impression and action-to-click attribution processes. [3] tackled the delayed feedback issue by introducing an additional model that captures the conversion delay. [12] proposed a multi-touch attribution strategy to narrow the discrepancy between local and global attributions. [9] presented a simple approach to estimating CVR by finding hierarchical groups of the user, publisher, and advertiser features. [15] proposed a framework that combines natural language processing and dynamic transfer learning for CVR prediction.

3 PERFORMANCE SAFE PREDICTION

For RTB predictions, over-prediction is a much more common phenomena than under-prediction during evaluations due to attribution allocation between the third party company and the platform itself. RTB has huge supplies and a systematic over-prediction could quickly lead to winning a massive amount of low quality impressions and rapidly exhaust the campaign budgets. Therefore, it is the top priority for designing a prediction system that has no systematic over-prediction prone, namely *performance safety*. In this section, we first analyze the rationales for CVR over-prediction. Then, we briefly describe our system design philosophy to adaptively control and correct systematic over-prediction.

3.1 CVR Over-Prediction Rationales

3.1.1 No Empirical Lower Bound. Compared to clicks on an ad, conversions are much more strong signals reflecting users' interest and intent for given ads. The user could simply click on an ad out of curiosity without any subsequent steps, or even just by mistake. We have observed significant amount of ads clicks even when random ads are shown in viewable publisher placements. Thus in practice it's safe to assume an empirical lower bound when predicting CTR. More specifically, let p_i be the true CTR for an impression, $\hat{p_i}$ the predicted CTR, and G_{cpc} the CPC goal amount. Ideally we should bid at the expected value $\hat{p}_i \cdot G_{\text{cpc}}$. In practice, we have observed that the clearing price is proportional to the bid price in RTBs that operate on second-price auction basis, that is, $\lambda \cdot p_i \cdot G_{\text{cpc}}$, where $0 < \lambda < 1$, then we can calculate the effective cost per click as:

$$eCPC = \frac{\sum_i \lambda \hat{p_i} \cdot G_{\text{cpc}}}{\sum_i p_i} = \lambda \cdot G_{\text{cpc}} \cdot \frac{\sum_i \hat{p_i}}{\sum_i p_i},$$

where the summations are over impressions. Due to the existence of the empirical lower bound for the true CTR p_i, as long as we don't over predict, i.e. $\hat{p_i} < p_i$, we are *performance safe*, meaning that we are guaranteed to achieve the CPC goal. If the empirical CTR lower bound is big enough to make a reasonable high bid price, we can safely do exploration without worrying about the performance sink. However, CVR is a completely different story; it's usually several magnitudes lower than CTR. And, unlike that for CTR, we don't observe any empirical lower bound for the true CVR. eCPA is calculated analogously as:

$$eCPA = \lambda \cdot G_{\text{cpa}} \cdot \frac{\sum_i \hat{p_i}}{\sum_i p_i},$$

where G_{cpa} is the goal amount. Note that the expected number of conversions $\sum_i p_i$ in the denominator could be arbitrarily close to 0 leading to the explosion of eCPA. Thus, the non-existence of CVR lower bound requires more effort to guarantee the performance safety.

3.1.2 Gap Between Observations and Predictions. As there is no way to observe each impression's true conversion rate, we can only compare predictions with observations from a group of impressions. We show that, under certain assumptions over-prediction is inherent even if our prediction is empirically unbiased.

Lemma 3.1. *Given n impressions, let $C_1, C_2, \ldots, C_n$ be their true conversion rates, and $\overline{C}$ their empirical conversion rate. Suppose that our predicted conversion rate $\widehat{C}$ is* unbiased *in that $\widehat{C} = \frac{1}{n}\sum_{i=1}^{n} C_i$, and we bid at the estimated true value $\widehat{C} \cdot G_{\text{cpa}}$. Furthermore, suppose that for each of these impressions the highest third-party bid price follows log-normal $\sim \ln \mathcal{N}(\mu_i, \sigma^2)$ with mean at the true value $\mu_i = C_i \cdot G_{\text{cpa}}$. Then $\widehat{C}$ is an over-prediction:*

$$\widehat{C} \geq \overline{C},$$

where the equality holds only if $C_1 = C_2 = \cdots = C_n$, that is, when all impressions under consideration have equal true conversion rates.

Proof. Let

$$w_i = \int_0^{\widehat{C} \cdot G_{\text{cpa}}} \ln \mathcal{N}(x; \mu_i, \sigma^2) dx$$

be the probability that the i-th impression wins by over-bidding all third-party platforms. Then the expected number of winning impressions is $\sum_{i=1}^{n} w_i$ and the expected number of conversions is $\sum_{i=1}^{n} w_i C_i$. Thus the observed empirical CVR is

$$\overline{C} = \frac{\sum_{i=1}^{n} w_i C_i}{\sum_{i=1}^{n} w_i}.$$

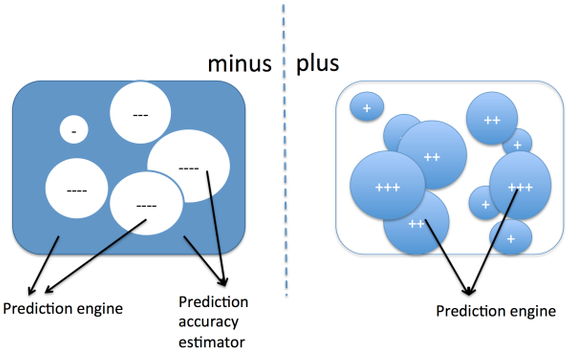

Figure 1: "Plus" v.s. "minus" training model approaches

By definition w_i can be written as

$$w_i = \Phi\left[\left(\ln\left(\widehat{C}\cdot G_{\text{cpa}}\right) - C_i \cdot G_{\text{cpa}}\right)/\sigma\right],$$

which is a decreasing function in C_i, where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution. Without loss of generality, assume that $C_1 \le C_2 \le \cdots \le C_n$. Then $w_1 \ge w_2 \ge \cdots \ge w_n$, and hence

$$\begin{aligned} n\widehat{C}\cdot\sum_{i=1}^{n} w_i &= \sum_{i=1}^{n} C_i \sum_{i=1}^{n} w_i \\ &= \langle C_1, C_2, \ldots, C_n\rangle\langle w_1, w_2, ..., w_{n-1}, w_n\rangle^T \\ &\quad + \langle C_1, C_2, \ldots, C_n\rangle\langle w_2, w_3, ..., w_n, w_1\rangle^T \\ &\quad + \cdots \\ &\quad + \langle C_1, C_2, ..., C_n\rangle\langle w_n, w_1, ..., w_{n-2}, w_{n-1}\rangle^T \\ &\ge n\cdot\langle C_1, C_2, ..., C_n\rangle\langle w_1, w_2, ..., w_{n-1}, w_n\rangle^T \qquad (1) \\ &= n\sum_{i=1}^{n} C_i w_i, \end{aligned}$$

where inequality (1) follows from the fact that, for any permutation $w_{(1)}, w_{(2)}, \ldots, w_{(n)}$, we must have $\sum_{i=1}^{n} w_{(i)}C_i \ge \sum_{i=1}^{n} w_i C_i$. It follows that $\widehat{C} \ge \overline{C}$. ∎

One key assumption in the above proof is that for an impression, there exists some other buying platform with average bidding price the same as ours, or *equilibrium conditions*, since all buying platforms are getting the same information as summarized in Subsection 1.1.

3.1.3 Training Data Limitations and Biases. First, the conversions are proprietary of advertisers, unlike clicks, when building a CVR prediction model for one advertiser, we have to exclude all other advertisers' data from training, which further limits the training examples. Second, the training data contains only RTB wins, which takes only a small proportion of entire RTB supplies, and this sampling is heavily biased by the previously serving model. There are some previous literature discussing how to handle these two limitations in practice [15, 17].

In our framework, we focus on alleviating the over-predictions caused by training sample selection bias. For example, for certain regions of the feature space where the previous model's prediction is suppressed, we get fewer winning instances. The few training examples could cause new models vastly over-predict in these regions. And after deploying into production, the new model then tends to bid aggressively but incorrectly over such regions. In the next section, we briefly describe our design philosophy.

3.2 Design Performance Safe Prediction System

With the presence of over-prediction, it is too risky to derive the bid price directly from the model's prediction without considering prediction qualities. There are generally two different approaches to make use of the prediction qualities. As illustrated in Figure 1, one is 'minus', the other is 'plus'.

(1) In the 'minus' way, two models are generated from training data, one outputs prediction values while another estimates prediction qualities. The final prediction is a function of the prediction discounted by its prediction qualities.
(2) In the 'plus' way, predictions initially are limited only to regions that have high confidence of their predictions. More regions are gradually added on to enable prediction in the model's life cycle when more feedback performance data are collected.

The 'minus' way requires to find all low prediction quality areas before deploying a model into production, which is impossible in practice. On the contrary, we are much safer to explicitly control the numbers and sizes of currently exploring areas with the 'plus' way. Thus, we adopted the 'plus' approach and have successfully implemented a novel system that continuously generate CVR predictions with high qualities. In Section 4, we describe details of this framework. Later, Section 5 presents some additional adjustments to further reduce the prediction biases.

4 CVR SAFE PREDICTION FRAMEWORK

In this section, we describe how to construct our performance-safe CVR prediction framework. We extend ideas similar to ensemble trees in the way that the final output ensembles predictions from a collection of trees, where each leaf node in a tree represents a subset of the feature space. And the major difference between our method and the regular ensemble tree method is that none of our trees is complete. Only leaf nodes with significant amount of historical data (or high-confident predictions in corresponding subsets) are generated and become active. Because trees in our framework are not complete, they are not static and keep growing during the campaigns' life cycles. With more data collected, new leaf nodes are created while some existing leaf nodes might be pruned. The growth of the trees is also carefully controlled to maintain performance safety of the overall prediction. To generalize, there are two types of trees in our system:

Data-Driven Trees are enumeration trees, where each leaf node in the same tree corresponds to a unique value combination from a common subset of features. The limitation of data-driven trees is that at the campaign starting time, they have no leaf nodes and therefore cannot bootstrap by themselves.

Machine-Learning Trees make predictions using machine learning models. Their main purpose is to help jump-start the data-driven trees' generation during the campaign initial stage.

Figure 2 illustrates how the trees evolved: initially there are no leaf nodes in the data-driven trees and only the machine-learning trees are capable of making predictions. As more past performance data are collected, the data-driven trees begin to grow and start to use leaf nodes to make more accurate predictions.

4.1 Data-Driven Tree

Each data-driven tree is an incomplete enumeration tree corresponding to a unique subset of features, and it tries to catch the interaction among the features. Figure 3 demonstrates an example of data-driven tree defined by the feature subset {`Gender`, `URL`}. It contains only four selected leaf nodes for demonstration, which is far fewer compared to the possible value combinations of these two features in our practice.

4.1.1 Construction. We leverage the capability of Gradient Boosted Decision Tree (GBDT) [5, 6] to identify strong feature interactions and

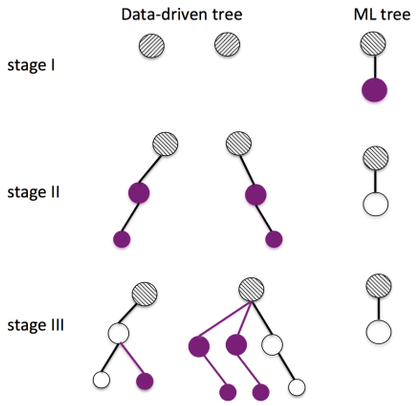

Figure 2: Tree evolution. Newly nodes are colored in purple.

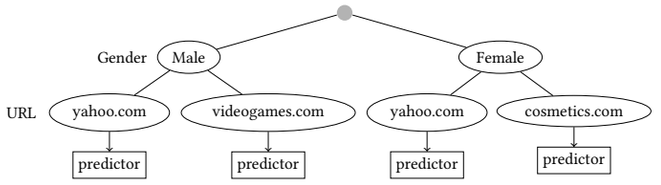

Figure 3: Data-Driven Tree

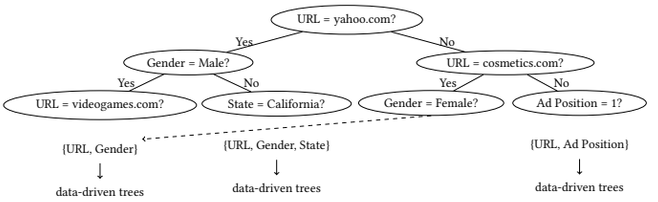

Figure 4: GBDT tree to Construct Data-Driven Trees

use these feature combinations to construct data-driven trees. We first train a GBDT model with data from all advertisers. Note that even though this GBDT model uses data across different advertisers, we do not use this model to predict CVR for each advertiser directly. Instead, this GBDT model is used only to guide us on how to create the data-driven trees. When the GBDT is trained, we extract the set of features based on the path from the root node to each leaf node, then a data-driven tree is built for each unique feature set. For example, assume the trained GBDT has a tree with the structure in Figure 4. Then the extracted feature sets are {`URL, Gender`}, {`URL, Gender, State`}, and {`URL, Ad Position`}, which are used to build the initial data-driven trees.

4.1.2 Prediction. A leaf node issues CVR predictions via the Beta-Binomial model, similarly to [1]. We assume that the number of conversions falling into the leaf node follows a binomial distribution. More specifically, given the CVR (or p) for the node that the impression is associated with, and the number of impressions (or B), the number of the resulted conversions A follows a binomial distribution $Binomial(B, p)$. We take a conjugate prior for p: $\mathbb{B}(\alpha, \beta) \stackrel{\text{def}}{=} Beta(\alpha, \beta - \alpha)$, with mean α and variance $\text{Var}(p) = \frac{\alpha(\beta-\alpha)}{\beta^2(\beta+1)}$. Here we defined $\mathbb{B}(\alpha, \beta)$ for convenience since α and β correspond to the *prior* conversion and impression counts respectively. The performance of this model is quite sensitive to the choices of α and β at the beginning [2]. A naive approach would be directly to set them to the conversion and impression numbers respectively at the beginning of the underlying CPA campaign. In other words, let A_t and B_t be the conversion and impression numbers at time t, then

$$\alpha \leftarrow A_t, \quad \beta \leftarrow B_t.$$

However, due to the dynamic nature of RTB, the true distribution is more likely to shift quickly over time; the naive estimates would be soon dominated by decayed data. A simple fix is to only use *recent* data. Let w be the preset length of a time window, say for example one month, then we use only data between time $t - w$ and t to estimate the parameters, that is,

$$\alpha \leftarrow A_t - A_{t-w}, \quad \beta \leftarrow B_t - B_{t-w}.$$

A more refined method is to use exponential decay over time, instead of the fixed-width time window. Let $0 < \delta < 1$ be the decay factor, then

$$\alpha \leftarrow \delta\alpha_{t-1} + (A_t - A_{t-1}), \quad \beta \leftarrow \delta\beta_{t-1} + (B_t - B_{t-1}).$$

The estimates using the decay factor would utilize more data but weigh more on recent data. In our framework, we take the exponential decay over time method as it outperforms the other two in terms of prediction accuracy in practice.

4.1.3 Tree Update. Initially all data-driven trees are empty at cold-start stage. New leaf nodes are added after sufficiently many impressions or conversions are observed. For example, the leaf node (`Genger=male`) $\times$ (`URL=yahoo.com`) is not in the tree until we have observed sufficiently many users who are `Male` and visited `yahoo.com`. More precisely, a leaf node is present in the tree only if the Beta distribution $\mathbb{B}(\tilde{\alpha}, \tilde{\beta})$ for the CVR satisfies

$$(\tilde{\alpha} > \alpha_0 \text{ or } \tilde{\beta} > \beta_0) \text{ and } \text{Var}(p) < v_0,$$

where α_0, β_0 and v_0 are preset thresholds according to CPA performance goals and $\tilde{\alpha}$, $\tilde{\beta}$ are posterior updates from the Beta-Binomial model. Based on these conditions, new leaf nodes may be added and disqualified leaf nodes may be pruned from the tree.

4.2 Machine-Learning Tree

Machine-learning trees are used to augment and bootstrap data-driven trees. When a new campaign is setup, there is no campaign-wise impressions or local attributed conversions so that the data-driven tree does not have any leaf node to make predictions. However, we can still collect its historical global conversions, since conversion pixels are required to fire before new campaign starts. Note that we could not directly feed these global conversions to data-driven trees. This is because not all of global conversions are related to impressions we have shown, and adding them into data-driven trees will introduce systematic bias to the CVR estimation at each leaf node. Instead, a machine learning model can take advantage of these global conversions by using them as additional positive examples during training. The trained machine learning model is able to predict user level conversion rate and is very suitable to jump-start a campaign. In our framework, machine-learning tree generates most cold-start CVR predictions and drive the growth of the data-driven trees. During campaign's flight time, the machine-learning model keeps being updated to takes into consideration of the most recent global conversion data.

To construct a machine-learning tree, we need to first define a confidence threshold. After that, a tree with a single leaf node is created. This leaf node will only provide predictions for any instance with score greater than the confidence threshold. In practice, for each GBDT model, we usually set the confidence threshold as the score cut-off of the top 10% scores obtained from the training dataset. 10% is chosen because it is big enough for the full delivery for most campaigns on our platform. Similarly to the data-driven tree, when a leaf node in the machine-learning tree accumulates sufficient data it could start to use the Beta-Binomial model for predictions, which is also dynamically updated throughout the campaign life cycle.

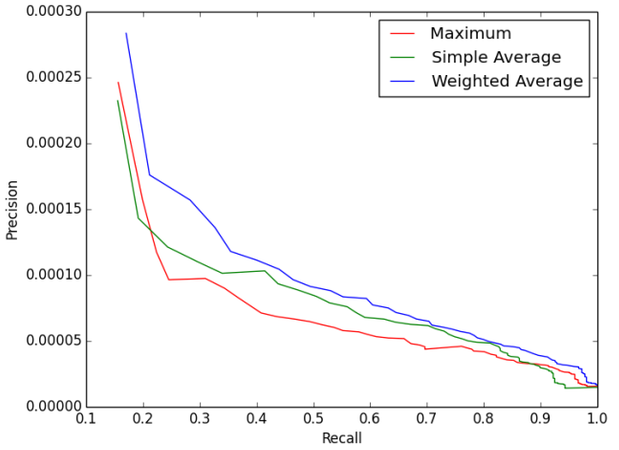

Figure 5: Comparing the precision-recall curve of the three ensemble method.

4.3 Ensemble Predictions

For each bid sample, each qualified data-driven or machine-learning tree makes a prediction on the CVR, then these predictions are combined to make the final prediction. We propose three different ensemble methods. Let $p_1, p_2, \ldots, p_n$ be the tree predictions.

Maximum The final prediction is the maximum of all predictions: $\hat{p} = \max\{p_1, p_2, \ldots, p_n\}$. The disadvantage is that it may lead to over-prediction.

Simple Average The final prediction is the average of all predictions: $\hat{p} = \frac{1}{n}\sum_{i=1}^{n} p_i$. The advantage is that it effectively prevents over-prediction. The disadvantage is that it assumes the underlying trees are equally accurate.

Weighted Average The final prediction is the average of all predictions weighted by their inverse variance: $\hat{p} = \frac{\sum_{i=1}^{n} \mathrm{Var}(p_i)^{-1} p_i}{\sum_{i=1}^{n} \mathrm{Var}(p_i)^{-1}}$. This method is applicable when the campaign is out of the cold-start stage, where the tree predictions follow Beta-Binomial models as described in Subsection 4.1.2. In other words, let $p_i \sim \mathbb{B}(\alpha_i, \beta_i) = Beta(\alpha_i, \beta_i - \alpha_i)$, then $\mathrm{Var}(p_i) = \frac{\alpha_i(\beta_i - \alpha_i)}{\beta_i^2(\beta_i + 1)}$.

Figure 5 compares their performance in terms of precision-recall. We used three days of impression data on 10 randomly selected campaigns with their labels defined as whether they can be attributed to some conversion events (positive sample) or not (negative sample). The weighted average method outperforms the other two options in general and we adopt it in practice.

5 CONVERSION ATTRIBUTION ADJUSTMENTS

In Section 4, we have introduced our CVR safe prediction framework, which heavily relies on the Beta-Binomial models used by all leaf nodes. However, due to some challenges mentioned before, a vanilla empirical mean is a biased estimation of the true CVR. In this section, we first discuss how to eliminate these biases, then show that to compute the true value of showing a CPA goal campaign ad, we also need to consider the probability of previous shown ad being the attributed ad. To the best of our knowledge, this practically important step has been neglected by previous publications.

5.1 Conversion Adjustment for Delayed Feedback

After showing an impression, a click usually happens within minutes. However, a conversion could lag in days or weeks to happen. Such a long delay makes it difficult to use recent impression data in prediction, since its corresponding conversion data has not completed yet. Blindly including these recent impression data into empirical estimation could lead to underestimate of the true CVR. One simple way to correct this bias is to discard the recent data from usage for a period. However, without considering recent data is detrimental to systems in dynamical environment like RTB. An alternative way is to keep using most recent data, but adding some estimations to compensate for the performance delay. We take the latter approach in our system.

For a given campaign, let's assume its attribution window is T days. Then on day i, we should wait for at least another T days to check whether there will be conversions in the following T days that can be attributed to impressions happening on day i. Therefore, if we use impression and conversion data in the last n days, where $n > T$, then only the impressions happening in the first $n - T$ days can get complete attributed conversions, but not those belonging to the last T days. Without careful consideration of this situation will lead to underestimation of the empirical CVR.

In practice, we find that both number of daily impressions and conversions have strong day-of-week pattern. Assume that an impression happens on day i, which is the d-th day of the week, $d \in \{0, 1, \ldots, 6\}$. Let k be the number of elapsed days between the conversion and the impression it is attributed to, $k \in \{0, 1, \cdots, T\}$. Then, define $P_d(k)$ as the impression attribution probability to a conversion that happens k days later, conditional on that there exists a conversion being attributed to this impression. Define the *fraction of attribution* as the percentage of how many conversions we can get from the log on day i

$$\alpha_i \stackrel{\text{def}}{=} \sum_{j:\, i \le j \le n} P_{D(i)}(j - i) = \sum_{k=0}^{n-i} P_{D(i)}(k),$$

where $D(i)$ is the ith day of a week Then α_i^{-1} can be used as a multiplier to correct the total number of conversions attributed to impressions on day i.

To calculate the fraction of attribution α_i, we need to estimate $P_d(k)$'s. One way is to use the empirical method. For i, j such that $1 \le i \le j \le n$, let $C_{i,j}$ be the observed number of conversions on day j being attributed to impressions on day i, then we estimate $P_d(k)$ as

$$\hat{P}_d(k) = \frac{\sum_{\substack{i \le n-T \\ D(i)=d}} C_{i,i+k}}{\sum_{k=0}^{T} \sum_{\substack{i \le n-T \\ D(i)=d}} C_{i,i+k}}.$$

The empirical method estimates $P_d(k)$ separately for $d \in \{0, 1, \ldots, d\}$ and it does not consider interactions between days of week. We adopt the following way to estimate $P_d(k)$ simultaneously. Given the number of conversions attributed to impressions on day i as $\sum_{j'=i}^{i+T} C_{i,j'}$ for all $i = 1, 2, \ldots, n - T$, the expected number of conversions on day j is

$$\sum_{i=j-T}^{j} \left(\sum_{j'=i}^{i+T} C_{i,j'} \right) \cdot P_{D(i)}(j - i),$$

for all $j = T + 1, T + 2, \ldots, n$. On the other hand, the observed number of conversions on day j is $\sum_{i=j-T}^{j} C_{i,j}$. Thus we can formulate the estimation as a constraint optimization problem to minimize the total squared errors between daily expected and observed conversions:

$$\min \sum_{j=T+1}^{n} \left[\sum_{i=j-T}^{j} \left(\sum_{j'=i}^{i+T} C_{i,j'} \right) \cdot P_{j-i,D(i)} - \sum_{i=j-T}^{j} C_{i,j} \right]^2$$

$$s.t.$$

$$\sum_{k=0}^{T} P_d(k) = 1 \text{ for } d = 0, 1, \ldots, 6,$$

$$0 \le P_d(k) \le 1 \text{ for } k = 0, 1, \ldots, T.$$

5.2 Conversion Adjustment for Local Attribution

As a buying platform, we can only do local attributions. There exists a discrepancy between local and third-party attribution numbers. The gap is generally too big to be ignored. If we know exactly which locally attributed conversion is also a third-party attributed one, we can easily build a classification model to directly predict the probability that a local attributed conversion is also a third-party attributed one. However, for privacy and other reasons, third-party only provides us aggregated data at campaign level on a daily basis.

Since we do not have granular enough data to predict probability for each conversion event, we first divide locally attributed conversions into groups. Each group is defined by a combination of factors that affect the third-party attribution probability. For example, one group is defined such that the conversion-impression elapsed time is one day and the user visited the advertiser's site 3 days ago. Let $g_1, g_2, \ldots, g_k$ be the predefined conversion groups. For each $i = 1, 2, \ldots, k$, let P_{g_i} be the probability that a conversion in group g_i is also third-party attributed. Then given the number of locally attributed conversions in group g_i with conversion time on day j, the expected number of third-party attributed conversions on day j, denoted as $C_{g_i,j}$, is $\sum_{i=1}^{k} \left(P_{g_i} \cdot C_{g_i,j}\right)$. Let $C_T, C_{T+1}, \ldots, C_n$ be the actual daily numbers of third-party attributed conversions. We can estimate P_{g_i}'s by minimizing the total squared errors as:

$$\min \sum_{j=T}^{n} \left[\sum_{i=1}^{k} \left(P_{g_i} \cdot C_{g_i,j}\right) - C_j \right]^2$$

$$s.t.$$

$$0 \le P_{g_i} \le 1 \text{ for } i = 1, 2, \ldots, k.$$

In practice, the values of P_{g_i}'s can be used to calculate the expected number of third-party attributed conversions, which can be in turn used to discount the CVR prediction.

5.3 Bid Price Adjustment

Given the goal amount G_{cpa} of the CPA campaign and conversion rate p_t at time t, ideally we should bid at the expected value of showing the impression:

$$V_{plain} = f(t) \cdot p_t \cdot G_{\text{cpa}},$$

where $f(t)$ is the probability that, given the conversion happens, it is attributed to the impression shown at time t. However, this value ignores an important fact: if the user was already shown an impression from the same campaign at an earlier time t_0, there is a *baseline* value even if we don't show the current impression:

$$V_{baseline} = f_{t_0}(t) \cdot p'_t \cdot G_{\text{cpa}},$$

where p'_t is the probability that the user will convert after time t without showing the new impression, and $f_{t_0}(t)$ is the probability that, given that the conversion happens, it will be attributed to the impression shown at time t_0.

Notice that p_t differs from p'_t only in that a new impression will be shown. For CPA campaigns a conversion requires stronger engagement from the user; the effect of a secondary impression is negligible. Hence we can use p_t as an approximation for p'_t, and the *incremental value* of the current impression can be calculated as:

$$\begin{aligned} \Delta V &= V_{plain} - V_{baseline} \\ &\approx f_t(t) \cdot p_t \cdot G_{\text{cpa}} - f_{t_0}(t) \cdot p_t \cdot G_{\text{cpa}} \\ &= [1 - f_{t_0}(t)/f_t(t)] \cdot V_{plain}. \end{aligned}$$

We call $1 - f_{t_0}(t)/f_t(t)$ the *value adjustment factor*. Our final bid price is then the incremental value, which is the value adjustment factor multiplying the CVR estimate from our model that has taken attribution probability into account.

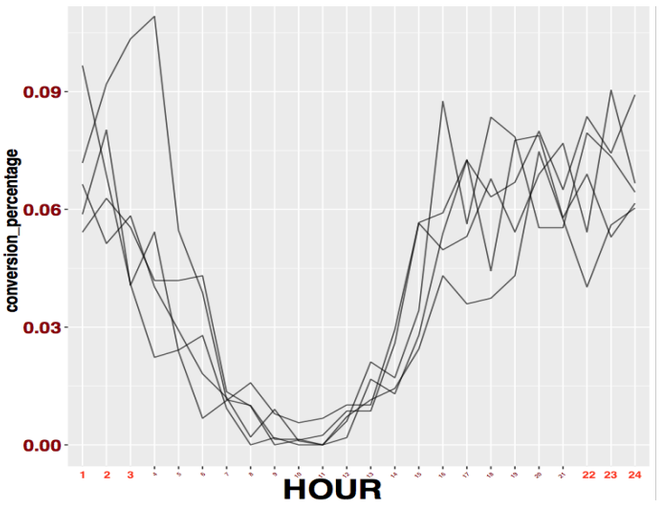

Figure 6: Campaign hour-of-day pattern

What remains is how to estimate the value adjustment factor. For any point of time $x > t$, let $\kappa(x)$ be the probability density that the conversion happens at x.Let q be the conditional probability that a conversion that happens at time x is attributed to the impression shown at time t_0. We observed in general that q becomes smaller if $x - t_0$ is larger. In practice we assume q is a function of $x - t_0$. Then

$$f_{t_0}(t) = \mathbb{E}_\kappa[q] = \int_t^{t_0+T} q(x - t_0) \cdot \kappa(x)\, dx.$$

Thus it's sufficient to estimate $\kappa(x)$ and $q(x - t_0)$. For $\kappa(x)$, we observed strong campaign-specific hour-of-day patterns as illustrated in Figure 6. For any time x within a predefined conversion window $[t, t + W]$, where $W < T$, we approximates $\kappa(x)$ as function of the hour of day at time x, denoted by $\text{hod}(x)$.

Algorithm 1 describes the steps to estimate the incremental value. The basic idea is to discretize the historical data and estimate the hourly probability mass of $\kappa(x)$ and $q(x - t_0)$, which are then used to calculate the final estimation for the value adjustment factor.

6 EXPERIMENTS AND EVALUATIONS

In this section, we describe our experimental results that demonstrate the effectiveness of the proposed CVR safe prediction framework and the conversion attribution adjustment models.

6.1 CVR Safe Prediction Framework

6.1.1 Render Feature Combinations to Create Data-Driven Trees. Each data-driven tree uses a unique subset of features to generate its leaf nodes. Feature subsets with strong capability to separate conversions from non-conversions are preferred to be selected. To achieve this, we trained a GBDT model with 2,000 trees and 562 categorical features. A constraint was added during the training to make sure no tree has more than 8 internal nodes, which implies that each tree has a maximum depth of 7 and each leaf node is defined by no more than 7 unique features. From the output of the trained GBDT model, feature combinations along with every path linking to a leaf node from its root node were identified. After removing duplications, 214 feature sets are obtained and each is used to create a corresponding data-driven tree in our system.

6.1.2 Prediction Accuracy of Data-Driven Trees. We evaluate the accuracy of data-driven trees by comparing the actually observed CVRs to those predicted by the data-driven trees. We set up 10 different campaigns and logged the final (unadjusted) predictions for 7 days for each impression.

Algorithm 1 Value Adjustment Factor Estimation

Input: time related parameters:
t_0 : time the previous impression was shown;
t : the current time to bid for a new impression;
T : length of attribution window (in hours);
W : length of conversion window (in hours).
Input: n hours of historical data.

1: For $i, j \in \{1, 2, 3 \ldots, n\}$, find counts $C_{i,j}$, the number of conversions from hour j that are attributed to impressions shown at hour i.
2: Calculate the probability mass of $\kappa(x)$ at hour $h \le W$:

$$\hat{\kappa}(h) = \frac{\sum^{n}_{\substack{j=T+1 \\ \text{hod}(j)=\text{hod}(h)}} \sum_{i=1}^{j} C_{i,j}}{\sum_{j=T+1}^{n} \sum_{i=1}^{j} C_{i,j}} \Big/ \left|\left\{h' \le W : \text{hod}(h') = \text{hod}(h)\right\}\right|,$$

and we set $\hat{\kappa}(h) = 0$ for $h > W$. Note that we excluded the first T hours of conversions since their attributed impressions may be incomplete.
3: Calculate the probability mass of $q(x - t_0)$ at hour h:

$$\hat{q}(h - t_0) = \frac{\sum^{n}_{\substack{j=T+1 \\ j>h-t_0}} C_{j-(h-t_0),j}}{\sum_{j=T+1}^{n} \sum_{i=1}^{j} C_{i,j}}.$$

4: Calculate attribution probabilities:

$$\hat{f}_{t_0}(t) = \sum_{h=t}^{t_0+T} \hat{q}(h - t_0) \cdot \hat{\kappa}(h), \text{ and } \hat{f}_t(t) = \sum_{h=t}^{t+T} \hat{q}(h - t_0) \cdot \hat{\kappa}(h).$$

Output: estimated value adjustment factor

$$1 - \frac{\hat{f}_{t_0}(t)}{\hat{f}_t(t)} = 1 - \frac{\sum_{h=t}^{t_0+T} \hat{q}(h - t_0) \cdot \hat{\kappa}(h)}{\sum_{h=t}^{t+T} \hat{q}(h - t_0) \cdot \hat{\kappa}(h)}.$$

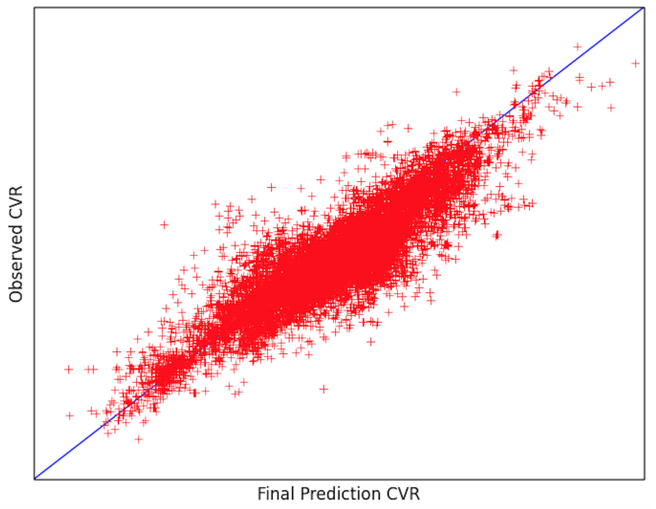

Figure 7: Final CVR predictions vs observed CVRs for one campaign. The coordinates are indexed by the negative logarithm of the value.

To evaluate the prediction accuracy of data-driven trees at impression-level, we can simply plot the observed CVRs against predicted CVRs. Figure 7 shows the result for one campaign, where the x-axis is the final predicted CVR and the y-axis is the observed CVR. We can see from the plot, the unadjusted CVRs are mostly over-predicting, which is an evidence that the adjustments explained in Section 5 are necessary.

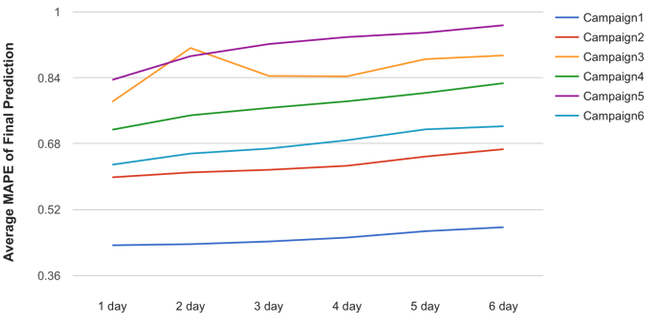

Figure 8: Predictions become increasingly inaccurate if leaf nodes are not updated with past-performance feedback loop.

For data-driven trees, frequent updates at each leaf node with most recent data are crucial to improve their prediction accuracy, especially in dynamical environment like RTB. Figure 8 illustrates that the prediction accuracies quickly drop (or the mean absolute percentage errors (MAPE) go up) if the leaf nodes are not updated with new data.

Table 1 shows that using data-driven trees and machine-learning trees together significantly improves both the performance and delivery of a campaign, as compared to using machine-learning trees alone. The test was run in production for two weeks with three test campaigns. For each campaign, the incoming traffic is randomly split into 50% as control and 50% as test. The control split uses only machine-learning trees and the test split uses both machine-learning trees and data-driven trees.

6.1.3 Effectiveness of Machine-Learning Trees. In our prediction framework, machine-learning trees bootstrap a campaign's delivery at cold-start time. It also grows up data-driven trees during a campaign's flight. The best way to measure machine-learning tree's effectiveness is to run online AB testing. However, data-driven trees cannot grow by themselves without resorting to external help. To handle it, in control part of our experiment, we substitute machine-learning tree's leaf node with one node that randomly select 10% of users. It is used to simulate the baseline scenario where machine-learning tree is not included. In Table 2, we compared results from 4 testing campaigns. For all of them, test parts exceed control parts in terms of both deliveries and performances. There are multiple reasons for it. First, comparing to random exploration, machine-learning leaf node has higher CVR and results more delivery by itself. Second, impressions generated by a random exploration leaf node are more likely to evenly fall into different data-driven tree leaf nodes instead of being concentrated into a few leaf nodes with high CVR predictions. This slows down the leaf node growing process and hinders the campaign's overall delivery. Machine-learning trees also bring significant performance lift, which illustrates that more efficient initial exploration can help to boost more higher CVR leaf nodes growth in data-driven trees.

6.2 Conversion Attribution Adjustment

6.2.1 Adjustment for Delayed Feedback. There are three lines in Figure 9. Each point in the triangle marker line represents the total number of conversions that can be locally attributed to event(s) on day i, $i = 1, 2, \ldots, 7$. In the Rhombus marker line, each point is the number of conversions that can be locally attributed to event(s) at day i with the observing time in the end of day 7. Points in the square marker line are estimated daily attributed conversion numbers via adjustment approach proposed in Subsection 5.1 by assuming that the time to do estimation is in the end of day 7. Figure 9 shows that after the adjustments, the estimated numbers are much closer to the actual ones, this is due to the reason that our model has accounted for the attributed conversions that have not been observed at the estimation time. After testing on 4

Campaign	Machine-Learning Tree			Machine-Learning + Data-Driven Tree			CVR lift	eCPA drop	imps lift
	CVR	eCPA	imps	CVR	eCPA	imps			
Campaign1	6.96E-06	63.5419	1,174,131	7.06E-06	61.9564	1,293,526	1.52%	-2.50%	10.17%
Campaign2	5.38E-06	130.2857	1,300,893	9.66E-06	83.8571	1,448,639	79.60%	-35.64%	11.36%
Campaign3	1.49E-03	0.8359	576,431	2.13E-03	0.5987	678,384	42.78%	-28.37%	17.69%

Table 1: Online test results from 3 campaigns. It shows performance and delivery improvements after adding data-driven trees on top of machine-learning trees.

	Imps lift	Convs lift	CVR lift
Campaign 1	+10.0%	+20.8%	+9.7%
Campaign 2	+4.7%	+7.5%	+2.7%
Campaign 3	+21.3%	+31.0%	+8.0%
Campaign 4	+1.5%	+2.1%	+0.5%

Table 2: Delivery and performance lift results from online A/B test by comparing using the machine-learning leaf node to bootstrap campaign's delivery with using a random learning leaf node.

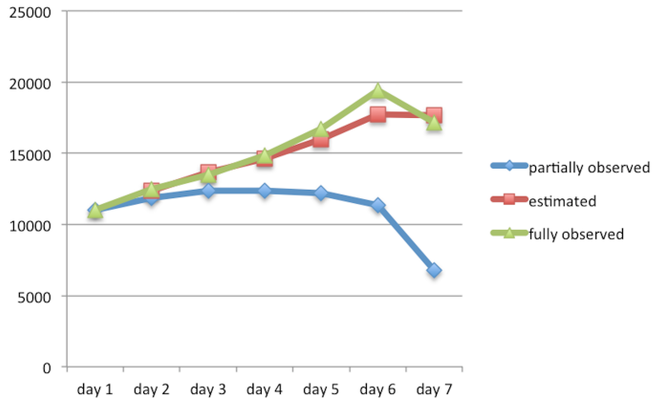

Figure 9: Comparison of the fully observed, partially observed, and estimated daily attributed conversion numbers.

	Campaign 1	Campaign 2	Campaign 3	Campaign 4
empirical	26.15%	26.25%	25.38%	14.17%
proposed	3.27%	1.96%	4.81%	6.09%

Table 3: MAPE comparison between empirical estimations and results from proposed optimization based approach.

campaigns for 14 days, as shown in Table 3, estimations with constraint optimization as proposed in Section 5.1 reduce the MAPE from 22.98% to 4.03% compared to the empirical estimation based adjustments.

6.2.2 Adjustment for Local Attribution. The experiment was set up as following. First, a testing campaign is selected and previous three weeks of data is used to formulate the allocation problem as described in Subsection 5.2. For this campaign, local attributed conversions are divided into 28 groups based on following three categorical variables:

- Elapsed time between the conversion and its local attribution. This variable has been converted into a categorical one with seven buckets from day 1 to day 7.
- A binary variable indicates whether the user has visited any web pages belonging to the same advertiser before the conversion happens.
- A binary variable denotes whether the conversion has been locally attributed to a click or an impression.

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6
Third-party Attr. #	48	39	66	24	17	11
Local Attr. #	88	53	90	51	53	22
Est. # using Avg.	53	32	54	31	31	13
Est. # using Model	50	28	53	29	27	13
Error Chg%	-8.3%	+12.8%	+1.5%	-8.3%	-23.5%	0.0%

Table 4: Model estimated conversion numbers vs. empirical estimated conversion numbers for a testing campaign in 6 days. The result shows 2% error rate reduction on average.

The model learned different third-party attribution probability for different groups. For this testing campaign, the average ratio between third-party and local attributions is 60.67%. And, the probabilities obtained from our model for different groups vary from 21% to 77%. In Table 4, *Third-party Attr.* row represents the number of third-party attributed conversions for this campaign at every day. *Local Attr.* row shows number of conversions happening and being locally attributed every day. *Est. # using Avg.* row displays the expected daily third-party attribution numbers with the average third-party attribution ratio 60.67%. *Est. # using Model* row shows the estimated number by using proposed algorithm in Subsection 5.2. Finally, *Error Chg* row summarizes the error differences by comparing results between row *Est. # using Avg.* and row *Est. # using Model*. On average, our proposed method reduces the estimation error by 2% througout the 6 days.

6.2.3 Bid Price Adjustment. Overall, we have observed around 5.6% improvement on return on investment (ROI) after performing one month A/B testing in our buying platform. In the test, we randomly split incoming bid requests into control and test groups, where only the test group bids applies value adjustment factors. Since the test group bids lower, we also need to measure the *attribution risk*, the discrepancy between local attribution and global attribution. High attribution risk would lead to high inaccuracy in local attributions and subsequently low model performances. Ideally, we should directly calculate the rate of local attributions that are not global attributions. However, we do not have third-party attribution data at impression level. Thus we use the elapsed time between a conversion and the last impression as a *proxy* to the *attribution risk*: the smaller the elapsed time, the more likely local and global attributions are consistent.

Figure 10 shows two examples illustrating how the distribution of elapsed time may change due to value adjustment factors. The left part shows that after applying the value adjustment factor we see more conversions with larger elapsed time in the test group, while the right part illustrates the opposite situation. Higher attribution risk exists for the left part. Table 5 compares the performance between the control and test groups. In addition to eCPA, we also compare delivery and attribution risk as well to have a complete picture. As we can see, after applying the value adjustment factors, we greatly reduced the inventory cost with the expense of very little delivery drop. Furthermore, the attribution risk actually becomes smaller.

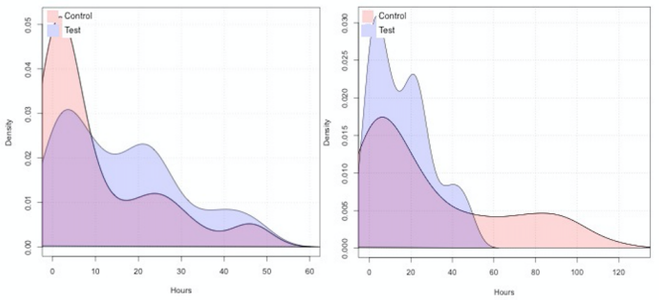

Figure 10: Examples of distribution change in elapsed time

Test Period	eCPA Drop	Delivery Drop	Attri Risk Drop
1st Week	5.06%	3.94%	5.42%
2nd Week	9.74%	1.68%	11.67%
3rd Week	8.70%	2.41%	9.03%
4th Week	9.26%	2.97%	5.47%

Table 5: Test results for bid price adjustment

7 CONCLUSIONS

Recently, more performance based digital advertising campaigns are choosing CPA as their goals. However, there is far less literature focusing on the differences between CVR and CTR predictions. Is CVR prediction really a natural extension of CTR prediction? In this paper, we offer an extensive analysis of the unique challenges faced by CVR predictions in the RTB environment. We introduce our safe CVR prediction framework which has been deployed at Yahoo! advertisement buying platform, with a particular focus on overcoming the hurdle of over predictions. Over predictions easily occur in high variance areas with rare events, which is a common practice in RTB environment. Unlike existing literature, which mainly uses cross validations to estimate the offline variance during model training, we rely more on evolving controlled explorations and real time feedback to more accurately estimate prediction variances. Conversion attribution adjustments are proposed and can help further alleviate over-bidding at different levels. We illustrate both offline and online experimental results to demonstrate the effectiveness of the framework.

In conversion predictions at RTB, there are a number of other challenges. First, generating training data set only from RTB winning impressions creates a huge selection bias for model training. Second, to simplify runtime system's complexity, we need to be able to estimate prediction variance reliably before deploying online prediction models. Third, while new model getting created frequently in the dynamical environment like RTB, its ability to transfer leanings from old model, especially to those areas that old model performs well, is crucial to ensure that the performance of the system continues to improve. We do not elaborate all these challenges in this paper, because they are not unique to conversion prediction problems. However, in any practical system, these challenges must be seriously considered and properly addressed. The focus on the current work is not to come up with the optimal solution to each of the above mentioned challenges, but try to highlight the problems as well as methods we have taken in practice. By doing so, we are hoping that there could be more research interests arising from machine learning community to help solve these real world challenges faced by display advertising industry.

REFERENCES

[1] Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. 2009. Spatio-temporal models for estimating click-through rate. In *Proceedings of the 18th international conference on World wide web*. ACM, 21–30.

[2] Peter Auer. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3, Nov (2002), 397–422.

[3] Olivier Chapelle. 2014. Modeling delayed feedback in display advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1097–1105.

[4] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. 2014. Simple and Scalable Response Prediction for Display Advertising. *ACM Trans. Intell. Syst. Technol.* 5, 4, Article 61 (Dec. 2014), 34 pages. DOI: http://dx.doi.org/10.1145/2532128

[5] Jerome H. Friedman. 2000. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics* 29 (2000), 1189–1232.

[6] Jerome H. Friedman. 2002. Stochastic Gradient Boosting. *Comput. Stat. Data Anal.* 38, 4 (2002), 367–378.

[7] Thore Graepel, Joaquin Quionero Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-Scale Bayesian Click-Through Rate Prediction for Sponsored Search Advertising in Microsoftfis Bing Search Engine. In *Proceedings of the 27th International Conference on Machine Learning ICML 2010, Invited Applications Track (unreviewed, to appear)*.

[8] Yu (Jeffrey) Hu, Jiwoong Shin, and Zhulei Tang. 2010. Pricing of Online Advertising: Cost-per-Click-through vs. Cost-per-Action. In *Proceedings of the 43rd Hawaii International Conference on System Sciences*. IEEE.

[9] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. Estimating Conversion Rate in Display Advertising from Past Erformance Data. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, New York, NY, USA, 768–776. DOI: http://dx.doi.org/10.1145/2339530.2339651

[10] Mohammad Mahdian and Kerem Tomak. 2007. Pay-per-action Model for Online Advertising. In *Proceedings of the 1st International Workshop on Data Mining and Audience Intelligence for Advertising (ADKDD '07)*. ACM, New York, NY, USA, 1–6. DOI: http://dx.doi.org/10.1145/1348599.1348600

[11] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting Clicks: Estimating the Click-through Rate for New Ads. In *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*. ACM, 521–530. DOI: http://dx.doi.org/10.1145/1242572.1242643

[12] Xuhui Shao and Lexin Li. 2011. Data-driven Multi-touch Attribution Models. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*. ACM, New York, NY, USA, 258–264. DOI: http://dx.doi.org/10.1145/2020408.2020453

[13] Robert Weber. 2003. Auction Theory: By Vijay Krishna. Academic Press, 2002. *Games and Economic Behavior* 45, 2 (2003), 488–497. http://EconPapers.repec.org/RePEc:eee:gamebe:v:45:y:2003:i:2:p:488-497

[14] Ling Yan, Wu jun Li, Gui rong Xue, and Dingyi Han. 2014. Coupled Group Lasso for Web-Scale CTR Prediction in Display Advertising. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, Tony Jebara and Eric P. Xing (Eds.). JMLR Workshop and Conference Proceedings, 802–810.

[15] Hongxia Yang, Quan Lu, Angus Xianen Qiu, and Chun Han. 2016. Large Scale CVR Prediction through Dynamic Transfer Learning of Global and Local Features. In *Proceedings of the 5th International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*. 103–119.

[16] Weinan Zhang, Shuai Yuan, and Jun Wang. 2014. Optimal Real-time Bidding for Display Advertising. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, New York, NY, USA, 1077–1086. DOI: http://dx.doi.org/10.1145/2623330.2623633

[17] Weinan Zhang, Tianxiong Zhou, Jun Wang, and Jian Xu. 2016. Bid-aware Gradient Descent for Unbiased Learning with Censored Data in Display Advertising. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, 665–674.