Gradient Boosting Censored Regression for Winning Price Prediction in the Real-Time Bidding

Piyush Paliwal and Oleksii Renov

LoopMe Ltd, London, United Kingdom
{piyush.paliwal,alexey}@loopme.com

Abstract The demand-side platform (DSP) is a technological ingredient that fits into the larger real-time-bidding (RTB) ecosystem. DSPs enable advertisers to purchase ad impressions from a wide range of ad slots, generally via a second-price auction mechanism. In this aspect, predicting the auction winning price notably enhances the decision for placing the right bid value to win the auction and helps with the advertiser's campaign planning and traffic reallocation between campaigns. This is a difficult task because the observed winning price distribution is biased due to censorship; the DSP only observes the win price in case of winning the auction. For losing bids, the win price remains censored. Erstwhile, there has been little work that utilizes censored information in the learning process. In this article, we generalize the winning price model to incorporate a gradient boosting framework adapted to learn from both observed and censored data. Experiments show that our approach yields the hypothesized boost in predictive performance in comparison to classic linear censored regression.

Keywords: Demand-Side Platform · Real-Time Bidding · Display Advertising · Censored Regression · Gradient Boosting.

1 Introduction

With growing popularity and usage [8], real-time bidding (RTB) has monetized the ad tech industry to a new scale. In the RTB auction process, buying and selling of online ad impressions are facilitated through an ad exchange in real-time. Whenever a visitor lands on a publisher's webpage or app, an ad request is initiated for a particular ad slot. An instant auction is invoked at the ad exchange wherein participating ad vendors (advertisers) can place their bids. The demand-side platform (DSP) assists advertisers to gainfully set a bid value on an impression, based on how relevant the user is towards the potential ad to be served. Several advertisers are bidding with the help of DSPs acting in their behalf. The advertiser with the highest bid value wins the auction and pays the price set by second highest paying advertiser. The winner's ad is then displayed on the publisher's webpage or app. The entire auction process finishes in under 100 milliseconds, hence the name real-time emerged.

From the advertiser's perspective, estimating the cost of winning the bid (aka the winning price) is profoundly important and guarantees the well-being of the ad inventory. Usually, the advertiser has a predefined campaign budget and the goal is to win ad impressions possessing the potential of returning better key performance metrics, for example, click-through rate (CTR) or conversion rate (CVR). Hence, the advertiser should bid higher on the ad impression with higher user response likelihood (CTR or CVR). Within this, DSPs are bidding on behalf of their advertisers. They use pre-set buying parameters to determine a value for each incoming ad impression. Any particular DSP has to have at least a shallow estimation of their competitors' bid price distributions. Suppose there are three DSPs, D1, D2, and D3, each bidding for the same ad request, say, with price equal to 10, 20 and 30 respectively. For D3, the winning price is the highest price set by its competitors, i.e., 20, which is set by D2. For D2, the winning price is 30, which is the highest price and set by D3. None of the DSPs has information about what their competitors are bidding. This is where predicting the winning price becomes necessary. For example [9] describes how bidding value is determined by a linear function of predicted CTR of the ad impression. [18] improved this computation further by forming a non-linear function of predicted CTR.

Usually the DSP has several campaigns with different targeting attributes which are competing between each other to show their ads on a specific bid request. The ability to predict winning price can help to reallocate requests possessing smaller win price on the ads with low bid price and requests possessing higher win price on the ads with high bid price. Another important application is to understand how many impressions, a campaign can buy in a desired time frame with the specified bid price [12].

In the literature, there are further works regarding the ad impression's winning price prediction using its features in a more direct way. However, data censorship makes this difficult. DSPs only observe the winning price of the bids which they win in auction. In the case when they lose the auction, the winning price is censored, and consequently unobserved. The full winning price distribution consists of the winning prices of all win bids plus the winning prices of all lose bids. Since the latter is missing in the training data, it becomes substantially challenging to model the winning price. However, in case of losing the auction, the DSP at least knows that the winning price is higher than its own bidding price. That is to say, the DSP knows a lower bound of the winning price for lost bids. Commonly, censoring is referred to as left, interval or right [3]. Left censoring occurs when observations are censored below the censoring point. Interval censoring occurs when observations are censored in between lower and upper censoring points. In right censoring, observations are censored above the censoring point. In our case, the bid price is right censored because the actual winning price is above this value.

We generalize the winning price model to incorporate a gradient boosting framework [4] adapted to learn from both observed and censored data. Boosting methods, in general, hold a dominant position in machine learning due to functioning very well in a wide range of applications. Gradient boosting combined with censored regression, together endeavor to mitigate the challenging task of winning price prediction. We approach two variants of gradient boosting with different weak learners, (i) using linear regression and, (ii) using decision tree, both adapted to censored regression. In [10] the authors define *grabit*, a gradient boosting decision tree adapted to censored regression for classification task. They formulate the loss function adapted to a censored model using left and right censoring. However, their work is for a significantly different application which predicts defaults on loans and both the right and left censoring points are static throughout the entire data. Our winning price prediction problem is directed toward right censoring only where the right censoring point, i.e. the lower bound of win price, is not static but varies for each ad request. Hence this requires a different definition of the loss functions, which we describe in Section 2. Few prior works have sought to deal with censorship for winning price prediction. [16] implements a linear censored model and [15] generalizes the winning price to use a deep learning framework to learn from censored data.

The contributions of this work are multi-fold. (1) We study current state-of-the-art winning price models. (2) We develop a new winning price algorithm which generalizes a gradient boosting framework to learn from both observed and censored data. (3) To the best of our knowledge, we are the first who utilize gradient boosting censored regression for the winning price prediction problem on the DSP side. (4) We experiment on the real and openly available iPinYou [19] dataset to evaluate our approach against the classic linear censored model.

2 Methodology

In this section, at first, we formally formulate the problem to be addressed, inspired by linear censored regression, which is a well known state-of-the-art algorithm. Next, we detail the mathematical background for our generalized winning price model that uses gradient boosting censored regression.

2.1 Problem Formulation

Suppose there are J DSPs connected to one ad exchange and bidding for incoming ad requests. We solve the task of predicting the auction price from the perspective of the k^{th} DSP. This means that we describe the modeling process from the standpoint of D_k. For the i^{th} bid represented by feature vector x_i, say, D_k is bidding with bidding price b_i. The features observed by the DSP usually accommodate information about publisher webpage or app, visiting user, and ad exchange. For D_k, the true winning price is the highest bidding price placed by its competitors, i.e. $D_1, \ldots, D_{k-1}, D_{k+1}, \ldots, D_J$. In an RTB auction, D_k has no idea of what the other DSPs are bidding.

As we introduced earlier, D_k can only observe the true win price in the case when it wins the auction. If another DSP wins a particular auction, then the only information which is known to D_k is the lower bound of win price, i.e., its own bid price denoted by b_i. Let us denote y_i as true win price and w_i as observed win price. Suppose that D_k bids higher than its competitors' bidding prices for the i^{th} ad request, then i^{th} bid wins the auction. In this case, the true win price equals the observed win price, i.e., $y_i = w_i$. If D_k bids lower than its competitors' bidding prices, the i^{th} bid loses the auction. That means b_i is $<$ unobserved win price. In this case, the true win price y_i is not directly observed by D_k.

In both cases, inspired by linear regression, we can approximate y_i based only on the available feature x_i as follows:

$$y_i = \theta^T x_i + \epsilon \tag{1}$$

Where $\theta \in \mathbb{R}^p$ and denotes a set of regression coefficients where p is the dimension of the feature vector x_i. Noise ϵ can be assumed to be independent and identically distributed from $N(0, \sigma^2)$ distribution. In such cases the true win price $y \sim N(\theta^T x, \sigma^2)$

Let $\mathcal{W}$ represents the set of all winning bids and $\mathcal{L}$ represents the set of all losing bids. Note that winning bids and observed data are used interchangeably, and so is losing bids and censored data.

Now, the likelihood of the winning price model on the observed data is simply the probability density function, denoted by ϕ, as follows:

$$\phi(\frac{w_i - \theta^T x_i}{\sigma}), \forall i \in \mathcal{W} \tag{2}$$

In the case of censored observations, the likelihood function can be expressed in terms of a cumulative density function, denoted by $\varPhi$. The reason why a DSP loses an auction is due to bidding lower than the actual unobserved win price. Hence we want to maximize the probability that the model will predict the win price above the bid price, aka the right censoring point, for lose bids.

$$P(y_i > b_i) = 1 - P(y_i \leq b_i), \forall i \in \mathcal{L} \tag{3}$$

By using Eq. (1):

$$P(y_i > b_i) = 1 - \varPhi(\frac{b_i - \theta^T x_i}{\sigma}), \forall i \in \mathcal{L} \tag{4}$$

Thus, the likelihood of the winning price model on the censored data is:

$$\varPhi(-(\frac{b_i - \theta^T x_i}{\sigma})), \forall i \in \mathcal{L} \tag{5}$$

By taking negative log and combining Eq. (2) and Eq. (5), we receive the overall negative log-likelihood function as follow:

$$\sum_{i \in \mathcal{W}} - \log \phi(\frac{w_i - \theta^T x_i}{\sigma}) + \sum_{i \in \mathcal{L}} - \log \varPhi(-(\frac{b_i - \theta^T x_i}{\sigma})) \tag{6}$$

The coefficient θ is learned by minimizing Eq. (6). This is commonly known as linear censored regression (LCR) in the state-of-the-art research [16] [1] [2]. This model learns from both the observed data $\mathcal{W}$ and the censored data $\mathcal{L}$.

2.2 Proposed Winning Price Model

We generalize the winning price model to incorporate a gradient boosting framework adapted to censored regression. The idea of boosting was first introduced by Friedman in [4]. Since then it has achieved a peak of success. We use xgboost [1], an optimized distributed gradient boosting library, and adapt it for use with our customized loss function as described in Subsection 2.2.2. Inspired by the library, **X***gboost*, and the idea of **C**ensored **R**egression, for the ease of exposition we call our winning price algorithm XCR.

2.2.1 Gradient Boosting

A boosting mechanism sequentially trains an ensemble of base learners in a forward stage-wise manner. In each stage, each base learner tries to compensate shortcomings (aka error residuals) made by previous learners. Following [4,5], the general boosting algorithm is briefly described in Algorithm 1.

Let (x,y) be input data of N samples, M be the number of boosting iterations and ρ be the step size. The set of functions $h(x;a_m)$ are base learners, which are learned sequentially using a forward stage-wise procedure. More specifically, at each stage m, $h(x;a_m)$ is chosen to minimize the loss function, L, using the negative gradient of L at the current model F_{m-1}.

Algorithm 1 Gradient Boosting

```
1: $F_0(x) = argmin_\rho \sum_{i=1}^{N} L(y_i, \rho)$
2: **for** $m = 1$ to $M$ **do**
3:     $\hat{y}_i = -[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{F(x)=F_{m-1}(x)}, \quad i = 1, \dots, N$
4:     $a_m = argmin_{a,\beta} \sum_{i=1}^{N} [\hat{y}_i - \beta h(x_i; a)]^2$
5:     $\rho_m = argmin_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \rho h(x_i; a_m))$
6:     $F_m(x) = F_{m-1}(x) + \rho_m h(x; a_m)$
7: **end for**
```

To incorporate learning from both the observed data and censored data, we define a custom loss function in Subsection 2.2.2 whose gradient and second order

[1] It is censored regression, however, as the mean of win price is assumed to depend linearly on x, we add the term *linear*. This helps to distinguish other versions of censored regression, for example, non-linear censored regression. The gradient boosting censored regression, which we introduce in this paper, is one example of non-linear censored regression.

[2] The likelihood Equation 5 in [16] has typo. Authors later added the Corrigendum: https://github.com/wush978/KDD2015wpp/blob/master/README.md.

derivatives are derived in Subsection 2.2.3. In xgboost, the Newton Raphson method [6] utilizes the gradient and second order derivatives to further minimize the loss function.

2.2.2 Loss Function for XCR

Inspired by linear censored regression, we generalize the loss function for gradient boosting censored regression. In the boosting method, the true win price $y \sim N(F(x), \sigma^2)$.

For simplicity we define:

$$z_i = \begin{cases} \frac{w_i - F(x_i)}{\sigma}, & i \in \mathcal{W} \\ \frac{b_i - F(x_i)}{\sigma}, & i \in \mathcal{L} \end{cases} \tag{7}$$

By replacing $\theta^T x_i$ with F(x_i) in the log-likelihood function defined in Eq. (6) and by using the definition in Eq. (7), we achieve a loss function for XCR as follow:

$$L(y, F(x)) = \sum_{i \in \mathcal{W}} - \log \phi(z_i) + \sum_{i \in \mathcal{L}} - \log \Phi(-z_i) \tag{8}$$

In boosting, the base learners are updated to improve the model, $F(x)$, at each step. Hence we derive the loss function w.r.t $F(x)$.

By using definition in Eq. (7) and following the definition of ϕ and Φ, let us define:

$$\frac{\partial \phi(z_i)}{\partial F(x_i)} = \frac{z_i}{\sigma} \phi(z_i) \tag{9}$$

and

$$\frac{\partial \Phi(-z_i)}{\partial F(x_i)} = \frac{\phi(z_i)}{\sigma} \tag{10}$$

2.2.3 Gradient and Second Order Derivative of Loss Function

By using definition in Eq. (8), the gradient, aka the first order derivative, for one particular observation (x_i, y_i) is:

$$\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = -\frac{\partial \log \phi(z_i)}{\partial F(x_i)} \mathbb{1}_{\{i \in \mathcal{W}\}} - \frac{\partial \log \Phi(-z_i)}{\partial F(x_i)} \mathbb{1}_{\{i \in \mathcal{L}\}} \tag{11}$$

By using Eq. (9) and Eq. (10):

$$\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = -\frac{z_i}{\sigma} \mathbb{1}_{\{i \in \mathcal{W}\}} - \frac{1}{\sigma} \frac{\phi(z_i)}{\Phi(-z_i)} \mathbb{1}_{\{i \in \mathcal{L}\}} \tag{12}$$

Next, the second order derivative is:

$$\frac{\partial^2 L(y_i, F(x_i))}{\partial^2 F(x_i)} = -\frac{\partial}{\partial F(x_i)}(\frac{z_i}{\sigma})\mathbb{1}_{\{i\in\mathcal{W}\}} - \frac{\partial}{\partial F(x_i)}(\frac{1}{\sigma}\frac{\phi(z_i)}{\Phi(-z_i)})\mathbb{1}_{\{i\in\mathcal{L}\}} \tag{13}$$

$$\frac{\partial^2 L(y_i, F(x_i))}{\partial^2 F(x_i)} = \frac{1}{\sigma^2}\mathbb{1}_{\{i\in\mathcal{W}\}} - \frac{\phi(z_i)}{\sigma^2\Phi^2(-z_i)}(z_i\Phi(-z_i) - \phi(z_i))\mathbb{1}_{\{i\in\mathcal{L}\}} \tag{14}$$

By using definition in Eq. (12), the second part of Eq. (14) can be expressed using only the first derivative:

$$\frac{\partial^2 L(y_i, F(x_i))}{\partial^2 F(x_i)} = \frac{1}{\sigma^2}\mathbb{1}_{\{i\in\mathcal{W}\}} + \frac{1}{\sigma}\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}(z_i + \sigma\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)})\mathbb{1}_{\{i\in\mathcal{L}\}} \tag{15}$$

3 Experiments

In this section, we investigate the effectiveness of the gradient boosting censored regression model (XCR) against the non-robust linear censored regression model (LCR). For LCR, we choose the Adam optimizer [7], an efficient algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. Adam has shown a benign convergence property in many applications. For XCR, we evaluate two variants of gradient boosting. The linear gradient boosting and tree gradient boosting both adapted to censored regression. For brevity, we refer to these two as XCR1 and XCR2, respectively.

In particular, we describe (i) the dataset preparation and split logic for the experiments, (ii) features used, (iii) evaluation metrics, (iv) hyper-parameters tuning using a validation set, and (v) detailed results showing how XCR outperforms LCR.

3.1 Dataset and Preparation Trick

We conduct and evaluate our algorithms on the iPinYou RTB dataset [3][19]. The dataset originally consists of three seasons of RTB records. Season 2 and season 3 occupy broader coverage of features than season 1. Hence we use the data of season 2 and season 3. These two seasons' data are also more popular and, hence, have been used in research like [16,15]. Both seasons contain bid request logs for 7 and 9 consecutive days respectively [4]. Before we split the data into training and testing sets, the data preparation trick is introduced as follows.

As DSPs only observe the winning price of win bids, the winning price for lost bids is not available in the iPinYou dataset. While the censored regression paradigm can be trained by intermingling uncensored (win bids) and censored

[3] http://data.computational-advertising.org/

[4] The bid prices are in Renminbi currency.

Table 1. The statistics for the simulated dataset of iPinYou Season 2.

Day	**# of bids**	**# of win bids**	**WR**	**AWP**	**AWP_$\mathcal{W}$**	**AWP_$\mathcal{L}$**
2013-06-06	1,821,479	1,514,416	0.83	74.8649	52.4677	185.3269
2013-06-07	1,806,062	1,524,314	0.84	72.3127	51.1205	186.9674
2013-06-08	1,634,967	1,352,038	0.83	81.1431	58.4850	189.4200
2013-06-09	1,651,630	1,366,097	0.83	81.3166	58.9570	188.2934
2013-06-10	1,920,576	1,603,798	0.84	79.8357	58.9134	185.7621
2013-06-11	1,745,905	1,461,085	0.84	79.6226	58.9162	185.8431
2013-06-12	1,657,578	1,378,728	0.83	79.9969	58.8019	184.7920

Table 2. The statistics for the simulated dataset of iPinYou Season 3.

Day	**# of bids**	**# of win bids**	**WR**	**AWP**	**AWP_$\mathcal{W}$**	**AWP_$\mathcal{L}$**
2013-10-19	228,133	170,739	0.75	86.7368	50.8938	193.3647
2013-10-20	214,295	159,646	0.74	87.8986	51.9533	192.9054
2013-10-21	848,760	621,626	0.73	90.0886	51.4093	195.9472
2013-10-22	681,700	503,108	0.74	91.5107	52.8885	200.3125
2013-10-23	226,791	170,850	0.75	89.2730	53.4484	198.6853
2013-10-24	245,897	197,279	0.80	74.1881	45.5880	190.2397
2013-10-25	318,240	245,671	0.77	82.2756	49.1967	194.2589
2013-10-26	268,380	198,709	0.74	89.6127	49.1872	204.9104
2013-10-27	110,467	84,730	0.77	86.0095	48.4357	209.7080

(lose bids) data, we still need to evaluate the algorithm's effectiveness on a test set. Without knowing the ground truth of winning price in lose data, we can not evaluate performance. For the sake of evaluation, we apply the data preparation tick originally proposed by [16] and employed subsequently in [15]. This strategy uses the original win bids data consisting of only observed winning prices, on which we apply the preparation trick to produce simulated win bids $\mathcal{W}$ and simulated lose bids $\mathcal{L}$ as follows. The original bidding price is lowered by 50% and called the new bidding price. If the new bidding price is greater than the original winning price, the corresponding bid is a simulated win bid. If the new bidding price is less than original winning price, the corresponding bid is a simulated lose bid. This mechanism preserves the ground truth of real winning prices both in $\mathcal{W}$ and $\mathcal{L}$, and also introduces data censorship by forming $\mathcal{L}$. Censorship is simulated in the dataset as follows, for the training set we use winning prices only for $\mathcal{W}$ and the winning prices in $\mathcal{L}$ are treated as missing. For evaluation on the test set, we use the ground truth of real winning prices both in $\mathcal{W}$ and $\mathcal{L}$.

The statistics of the simulated data are shown in Table 1 and Table 2. In these tables, # of bids is the number of bids in the full simulated data and # of win bids is the number of winning bids in $\mathcal{W}$. WR represents the winning rate calculated by dividing # of win bids by # of bids. AWP is the average winning price on all simulated bids. AWP_$\mathcal{W}$ and AWP_$\mathcal{L}$ are the average winning price for $\mathcal{W}$ and $\mathcal{L}$, respectively.

We now randomly split the data, 90% to the training set and the remaining 10% to the testing set. We do this for each season and each day to produce: (i) for season 1, 7 new data collections which form 7 new training sets and 7 new testing sets, (ii) for season 2, 9 new data collections which form 9 new training sets and 9 new testing sets. We further randomly sample 10% from each training set to form a corresponding validation set to be used for hyperparameter tuning which is described in Subsection 3.4.

3.2 Features

Each bidding log has 26 attributes related to user info, ad exchange and ad slot etc. [16,19]. Most features are categorical. Some features such as Region and City are numeric in appearance but not in physical denotation. The features such as AdSlotWidth and AdSlotHeight are numeric not only in appearance but also in physical denotation. However, they represent finite levels which can be treated either by vectorizing into groups or simply as categorical. For simplicity, we process all of them as categorical. These features are then converted to binary features via the efficient hashing trick proposed by [14].

3.3 Evaluation Metric

Root mean squared error (RMSE) and mean absolute error (MAE) are commonly known metrics. RMSE is widely used as there is good reason to assume that noise follows a Gaussian distribution. While RMSE gives disproportionate weight to large errors, MAE weights equally all the individual absolute differences, therefore is less sensitive to outliers. The choice of metric depends on what is important for the given application. Since in our problem, the large errors are particularly undesirable, using RMSE is more valuable than MAE. The smaller the RMSE between true win prices and predicted win prices, the better the model.

3.4 Hyperparameter Tuning

The training set is used for learning and the validation set, with early stopping [17], is used to find an optimal set of parameters. The validation performance metric is RMSE on overall validation data.

Exhaustive search of the hyperparameters space is intractable. Therefore, we decided to limit the parameters to reasonable ranges. We use the bayesian optimisation techniques described in [11] for hyperparameter optimization of all models over these reasonable ranges. For LCR, we tune the learning rate and L_2 regularization. σ is defined as the standard deviation of the true win price. Inspired by [16], we compute σ as the standard deviation of the observed win prices in the training set. In both variants of XCR, XCR1 and XCR2, new

Table 3. Bayesian optimization experiments results on day 2013-06-10

Model Name	Validation RMSE with different configurations of parameters					
XCR2	**eta**	**max_depth**	**gamma**	**colsample**	**subsample**	**RMSE**
	0.169	65	11.0	0.83	1.0	37.38
	0.13	41	7.5	0.41	0.80	39.93
	0.08	54	8.3	0.81	0.40	40.17
XCR1	**eta**	**L_2 reg**	**RMSE**			
	0.09	0.2	39.57			
	0.05	0.5	40.60			
	0.15	1.0	42.66			
LCR	**eta**	**L_2 reg**	**RMSE**			
	0.03	0.5	39.98			
	0.1	0.8	42.50			
	0.08	0.61	42.63			

base learners are sequentially added to correct the errors made by the existing sequence of base learners. In XCR1, the underlying base learners are linear models, while XCR2 uses tree-based base learners. For both, we tune the learning rate which acts as a weighting factor for the corrections made by the new base learners. For XCR1, we additionally tune L_2 regularization to avoid overfitting. The learning rate, which is a common parameter in all three methods, is tuned in interval $[0.001, 0.5]$. The L_2 regularization, which is common in LCR and XCR1, is tuned in interval $[0.01, 5]$.

For XCR2, we additionally tune (i) the maximum tree depth in $[5, 70]$ (ii) the gamma parameter in $[0, 15]$ to avoid overfitting by adding more constraints on the partition of the leaves in a tree, and (iii) the data subsampling in $[0.2, 1]$ and feature sampling, in $[0.2, 1]$ for each base learner, so that base learners learn from each other in an optimal manner. Too large max depth generally can cause overfitting, but this is subjective and the trend varies across a different set of problems. For example, [13] achieves good results when a moderate max depth is set.

Table 3 presents a sample of optimization results for all three models on 2013-06-10. Some parameters' names are shortened. The learning rate, data subsampling and feature subsampling are denoted by *eta*, *subsample* and *colsample*, respectively.

When the tuning phase is finished, we build a model again using the optimized parameters running until full convergence is reached. If the model still

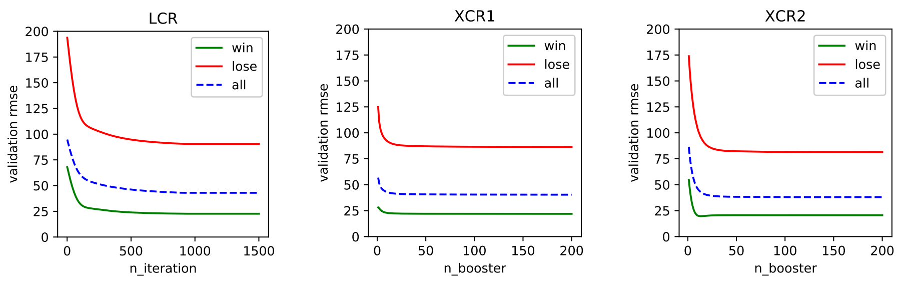

Fig. 1. Validation RMSE w.r.t iterations on win, lose and all bids of day 2013-06-10 for LCR, XCR1 and XCR2.

overfits, the training is stopped by early stopping. Fig. 1 [5] illustrates that smooth convergence for all three models is achieved according to validation RMSE overall, denoted by label *all* (the dotted blue line). For LCR, the number of iterations, denoted by *n_iteration*, is used. For XCR1 and XCR2, iterations correspond to the number of boosting updates, denoted by *n_booster*.

3.5 Evaluation Results

This subsection presents quantitative results to validate the effectiveness of our algorithm for winning price prediction. Overall results for each day are listed in Table 4 and Table 5 for season 2 test sets and season 3 test sets, respectively.

3.5.1 Discussion on Overall Results

The reported RMSE (error) shows that, for most days, XCR2 outperforms LCR and XCR1. This holds true for the win bids, the lose bids and overall. The tree based XCR2 is capable of learning the interactions between features, which is not possible for linear models LCR and XCR1. Although XCR1 also uses gradient boosting mechanism, but due to the underlying linear base learners, the end results are still linear. Hence XCR1 and LCR generally perform similar, if converged well.

As shown in Table 1 and Table 2, the amount of data in season 2 is significantly higher than in season 3. Consequently, for all three models, the performance on season 2 is better than on season 3.

3.5.2 Discussion on Skewed Winning Price Distribution

The results also show that, for all three models, the error on lose bids is significantly higher than the error on win bids. To explore the reason, we look at the

[5] We choose 2013-06-10 to illustrate the validation process, because this day has the highest number of bids as shown in Table 1. The overall results for each day on the testing set are discussed in Subsection 3.5.

Table 4. Root mean squared error on testing set of simulated dataset of iPinYou Season 2.

	RMSE all			**RMSE win**			**RMSE lose**		
Day	LCR	XCR1	XCR2	LCR	XCR1	XCR2	LCR	XCR1	XCR2
2013-06-06	45.59	46.44	**43.91**	25.63	25.59	**24.43**	95.57	98.02	**92.36**
2013-06-07	45.12	45.66	**43.19**	26.32	25.17	**23.58**	96.31	99.52	**94.45**
2013-06-08	47.06	47.84	**46.14**	24.53	24.47	**23.78**	99.35	101.53	**97.73**
2013-06-09	45.51	45.74	**43.56**	26.08	24.66	**23.53**	93.42	95.88	**91.25**
2013-06-10	41.31	40.41	**38.10**	22.03	21.91	**20.51**	88.86	86.43	**81.70**
2013-06-11	45.11	41.76	**41.20**	23.39	22.44	**20.65**	98.11	**89.83**	90.45
2013-06-12	41.19	42.06	**39.80**	24.27	22.61	**21.36**	85.22	89.95	**85.19**

Table 5. Root mean squared error on testing set of simulated dataset of iPinYou Season 3.

	RMSE all			**RMSE win**			**RMSE lose**		
Day	LCR	XCR1	XCR2	LCR	XCR1	XCR2	LCR	XCR1	XCR2
2013-10-19	59.61	60.09	**57.46**	33.96	34.34	**32.56**	103.29	104.04	**99.74**
2013-10-20	57.42	56.14	**55.34**	37.65	34.34	**33.98**	94.31	95.02	**93.52**
2013-10-21	57.48	55.97	**54.83**	40.15	35.36	**33.80**	**89.33**	91.30	90.33
2013-10-22	61.33	58.29	**57.27**	41.35	35.44	**34.22**	97.56	96.98	**95.90**
2013-10-23	59.36	59.07	**57.64**	38.88	37.24	**36.20**	98.23	99.45	**97.19**
2013-10-24	**54.46**	56.13	55.22	34.50	34.83	**34.05**	**101.44**	105.97	104.13
2013-10-25	56.42	57.89	**55.67**	34.90	35.95	**34.12**	98.42	100.81	**97.47**
2013-10-26	62.37	59.02	**56.56**	36.33	34.49	**33.84**	106.41	100.57	**95.60**
2013-10-27	61.61	58.24	**56.63**	34.88	32.40	**31.75**	110.54	105.09	**101.93**

average winning price on both winning and losing bids, represented by AWP_$\mathcal{W}$ and AWP_$\mathcal{L}$, respectively, as shown in Table 1 and Table 2. The AWP_$\mathcal{L}$ is usually higher than the AWP_$\mathcal{W}$ for both seasons. This is expected behavior as the DSP loses auctions by bidding lower than the win price. One can also see that on average the win rates, WR, are approximately 0.83 and 0.74 for season 2 and season 3, respectively. This introduces a skewed distribution biased more towards the win bids. This makes it more difficult for censored regression to predict lose bids as well in comparison to win bids. We see a similar trend in related research [16,15]. While predicting lose bids in general is difficult, XCR2 still delivers relatively better for lose bids in comparison to the other two models.

3.5.3 Discussion on Predictions

To illustrate the behavior of predictions, we choose the day 2013-06-10 from season 2. Fig. 2 compares densities of the predictions generated by all three methods against the density of true win prices in the testing set. We show this differently for the winning bids and the losing bids, appearing in the left and right

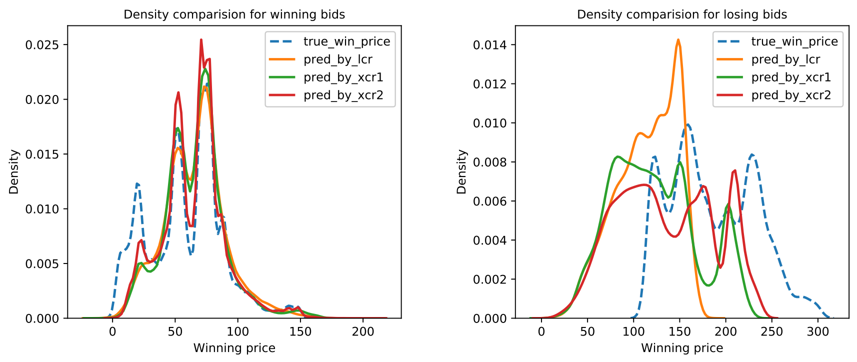

Fig. 2. Predictions densities of winning prices from all three methods against original winning prices on day 2013-06-10.

side of the plot, respectively. The label *true_win_price* represents the density of original win prices. The prediction densities of LCR, XCR1 and XCR2 are labelled by *pred_by_lcr*, *pred_by_xcr*1 and *pred_by_xcr*2, respectively.

For the win bids, the prediction density of XCR2 captures the true win price slightly better than the other two competing models (LCR and XCR1), as the red spikes match with the dotted blue more accurately. The other two models also reasonably capture the original true win price.

For the lose bids, the difference is more clearly visible. The prediction density of XCR2 captures the true win price considerably better than the other two models. The same can be seen from the overall results in terms of RMSE, as discussed in Subsection 3.5.1.

Another key advantage of using a tree-based boosting approach is to produce relatively stable predictions. In our application, the win price cannot sensibly take negative values, such a prediction is clearly incorrect. Linear models are prone to behave arbitrarily beyond the domain spanned by training examples. To circumvent this disadvantage, one can set a threshold such that if predicted win price is beyond that threshold, it is assigned the minimum/maximum acceptable value. This still requires manual adjustment and does not make predictions very accurate. Fig. 2 illustrates the linear models LCR and XCR1 not respecting the known bounds of win price, i.e., few predictions are negative. Although it is hard to see in the density plot because the overall contribution is small; LCR and XCR1 produce 0.062% and 0.02% negative predictions in the test data, respectively. In this particular example, XCR2 does not produce any negative predictions. Although the tree based boosting methods can also predict negative values, but the overall contribution is less than in the linear case.

To conclude, while the difference in error can vary depending on the day and part of the data (win or lose), overall XCR2 shows a better predictive power that enhances the performance of winning price prediction.

4 Related Work

We review the related work of studying winning price in real-time bidding. [2] has previously studied the prediction of winning prices using gradient boosted decision trees. However, they are on the seller side which does not have the problem of censored data.

In [16] the authors modeled the censoring of the winning price from the side of the ad impression buyer. They were the first to address the wining price problem on the DSP side with censored regression. They built a linear censored regression model, a linear uncensored regression model and a winning rate mixture model to predict the winning price. Our work generalizes the winning price model to incorporate a gradient boosting framework to learn from both observed win bids and censored lose bids. We do not yet employ a mixture model as they did and is still future work for us. For example, predictions that come from our model are learned from both win bids and lose bids. We could then learn a simpler uncensored model, which will be trained only on win bids, to weight predictions by winning rate to form the mixture model. Hence, this work extends flexibility in various ways. We have compared our reported RMSE to theirs on several days from both season 2 and season 3 (they actually used the mean squared error, MSE, in figures, but it is easier to compare on the same scale of RMSE). Our gradient boosting censored regression, particularly tree-based XCR2, outperformed their results of linear censored regression on most days.

[13] incorporated a censored model into decision tree learning to predict right-censored market price observations. We can not compare our result with theirs because their data simulation trick and reported evaluation metrics do not correspond to ours. They also reported results by different advertisers.

More recently, [15] generalized the winning price model to incorporate deep censored learning models with different distributions and network structures. They did not train and evaluate models on a daily basis like [16] and our work. They reported MSE and other metrics for season 2 and season 3 overall. We see that they achieved their best results under the wide and deep network structure when the deep censored method was trained on both observed and censored data. Our results of season 2 for a few days show better results than theirs for lose bids in terms of RMSE, but not overall. Although our results on win bids are not as good as their best achieved result, particularly for season 3, their results from other network structures are still close to ours. It is not possible to do a fair comparison as they tuned hyperparameters using the average log probability metric and the amount of data they used for training is 90% of entire season 2 and season 3, separately, while we train and evaluate on a daily basis to have a fair comparison against [16].

To our knowledge, we are the first to use a gradient boosting framework adapted to censored regression for predicting the winning price on the DSP side. We show that our approach can enhance the performance of a winning price model in comparison to classic linear censored regression.

5 Conclusions and Future Work

In this paper, we generalized the winning price model to incorporate a gradient boosting framework adapted to censored regression (XCR) in RTB display advertising. We evaluated two variants: XCR2 which uses tree base learners and XCR1 which uses linear base learners. We compared both variants against classic linear censored regression (LCR). Our findings reinforce the fact that the proposed gradient boosting censored regression (tree-based boosting XCR2) addresses the complexity of winning bid price determination in the RTB environment and demonstrates superiority over classic linear censored regression.

In future work, we intend to extend the flexibility of our model (i) by relaxing the normality assumption for winning prices, by using different distributions, and (ii) by forming a mixture model as discussed in Section 4. Moreover, it would also be a logical choice to combine our approach with a user response likelihood (e.g. click-through rate, conversion rate, etc.) model to further improve performance.

References

1. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794. KDD '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939785, http://doi.acm.org/10.1145/2939672.2939785
2. Cui, Y., Zhang, R., Li, W., Mao, J.: Bid landscape forecasting in online ad exchange marketplace. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 265–273. KDD '11, ACM, New York, NY, USA (2011). https://doi.org/10.1145/2020408.2020454, http://doi.acm.org/10.1145/2020408.2020454
3. Elandt-Johnson, R.C., Johnson, N.L.: Survival models and data analysis / Regina C. Elandt-Johnson, Norman L. Johnson. John Wiley and Sons New York (1980)
4. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (Aug 1997). https://doi.org/10.1006/jcss.1997.1504, http://dx.doi.org/10.1006/jcss.1997.1504
5. Friedman, J.: Greedy function approximation: A gradient boosting machine. The Annals of Statistics **29** (11 2000). https://doi.org/10.1214/aos/1013203451
6. Friedman, J., Hastie, T., Tibshirani, R., 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)
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014)
8. Muthukrishnan, S.: Ad exchanges: Research issues. In: Leonardi, S. (ed.) Internet and Network Economics. pp. 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
9. Perlich, C., Dalessandro, B., Hook, R., Stitelman, O., Raeder, T., Provost, F.: Bid optimizing and inventory scoring in targeted online advertising. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 804–812. KDD '12, ACM, New York, NY, USA (2012). https://doi.org/10.1145/2339530.2339655, http://doi.acm.org/10.1145/2339530.2339655

10. Sigrist, F., Hirnschall, C.: Grabit: Gradient tree boosted tobit models for default prediction (2017)
11. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2. pp. 2951–2959. NIPS'12, Curran Associates Inc., USA (2012), http://dl.acm.org/citation.cfm?id=2999325.2999464
12. Wang, J., Zhang, W., Yuan, S.: Display advertising with real-time bidding (RTB) and behavioural targeting. CoRR **abs/1610.03013** (2016), http://arxiv.org/abs/1610.03013
13. Wang, Y., Ren, K., Zhang, W., Wang, J., Yu, Y.: Functional bid landscape forecasting for display advertising. In: Frasconi, P., Landwehr, N., Manco, G., Vreeken, J. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 115–131. Springer International Publishing, Cham (2016)
14. Weinberger, K., Dasgupta, A., Langford, J., Smola, A., Attenberg, J.: Feature hashing for large scale multitask learning. In: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 1113–1120. ICML '09, ACM, New York, NY, USA (2009). https://doi.org/10.1145/1553374.1553516, http://doi.acm.org/10.1145/1553374.1553516
15. Wu, W., Yeh, M.Y., Chen, M.S.: Deep censored learning of the winning price in the real time bidding. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining. pp. 2526–2535. KDD '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3219819.3220066, http://doi.acm.org/10.1145/3219819.3220066
16. Wu, W.C.H., Yeh, M.Y., Chen, M.S.: Predicting winning price in real time bidding with censored data. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1305–1314. KDD '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2783258.2783276, http://doi.acm.org/10.1145/2783258.2783276
17. Zhang, T., Yu, B.: Boosting with early stopping : Convergence and consistency (2003)
18. Zhang, W., Yuan, S., Wang, J.: Optimal real-time bidding for display advertising. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1077–1086. KDD '14, ACM, New York, NY, USA (2014). https://doi.org/10.1145/2623330.2623633, http://doi.acm.org/10.1145/2623330.2623633
19. Zhang, W., Yuan, S., Wang, J., Shen, X.: Real-time bidding benchmarking with ipinyou dataset (2014)