Optimizing Advertiser Utility In Real Time Bidding

Jim Caine
November, 2014
DePaul University – CSC 594

1 Introduction

Real time bidding (RTB) for display media allows for publishers to sell individual impressions to advertisers as the ad space becomes available. This is in contrast with more traditional direct buys, where the publisher sells ad space to the advertiser ahead of time. For marketers, real time bidding allows advertising campaigns to be user-targeted, for purchase decisions to be made for each impression (instead of a block of impressions), and for the campaigns to be run amongst a diverse set of publishers. For publishers, real time bidding results in increased revenue because publishers can utilize the new venue for selling ad space when the price is right. Users also benefit from the exposure to more relevant advertisements.

The manner in which an advertiser navigates the ad exchange landscape and responds to bids is of great interest to advertisers. Optimally responding to ad exchange bids results in increased targeting accuracy and decreased prices for the advertiser, ultimately resulting in increased return on their advertising investment. There are two key components to optimally responding to bids. First, the quality of each impression must be valued to discriminate between good performing impressions and bad performing impressions. Lastly, bids must be scaled according to the landscape of the market to ensure that advertiser utility is being maximized given a budget – this is referred to as the bidding strategy throughout the paper.

In this analysis, historical RTB bidding data is used to simulate the performance of an advertising campaign. First, various machine learning algorithms are applied to the dataset to predict the propensity to click for any given impression. Next, three different bidding strategies are implemented: flat bidding, goal bidding, and dynamic bidding. The flat bidding approach is the most basic approach and used as a benchmark. Goal bidding should show increased performance over flat bidding as it takes advantage of the propensity to click algorithm that is created. Lastly, the dynamic bidding approach shows that the lift in performance that is seen in the goal bidding strategy can be seen in a budget-constrained campaign.

2 The Data

The dataset used in this analysis is derived from the iPinYou Global Bidding Algorithm Competition[1]. The dataset is a real dataset, created by running a series of actual campaigns with a flat 300 Yen bid. The dataset is made up of four different logs describing a set of advertising campaigns over a seven-day period: a bids log, impressions log, clicks log, and conversions log. Each log contains data for every bid, impression, click, and conversion event throughout the time frame. Each event contains data about the user such as their city, data about the ad space such as the size of the ad, and data about the bid such as bidding price (and paying price for impression log). Because the performance for impressions that were not won in the dataset cannot be evaluated (we will never know if an impression gets clicked if we are unable to serve the impression), the bids log is ignored. The clicks log is used in favor of the conversions log to represent events because the number of clicks outweighs the number of conversions.

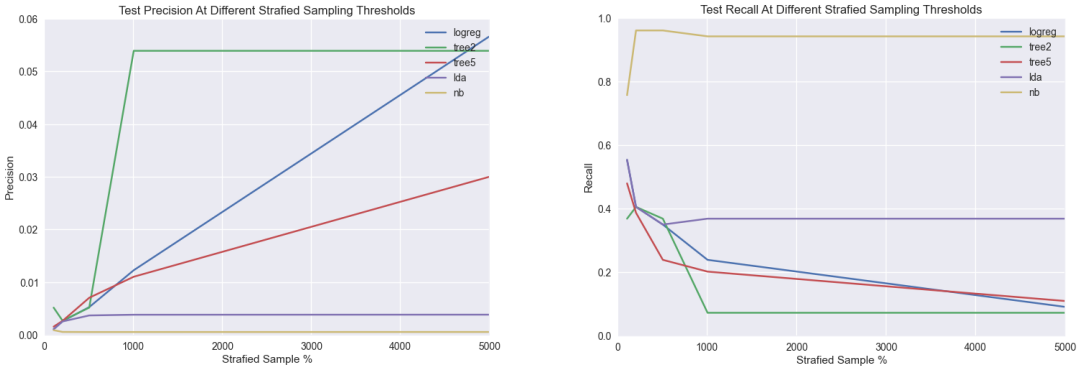
A training and test set of data is partitioned chronologically to mimic how the algorithm would be used in a real world setting. The logs for June 8, 2013 are used for the training set. The test set is comprised of a random sample of 100,000 impressions that occurred on the following day, June 9, 2013.

A series of preprocessing steps is applied to the data before being able to fit the model. First, the logs are parsed and only the most relevant attributes are selected for the model. Next, the impression and click logs are merged by searching through the click log for each impression for a click event by the same user within 2 minutes of the impression. A stratified sample of impressions resulting in clicks and impressions resulting in no click is taken at different stratified ratios. The attributes that are not common between the training and test sets are dropped, as attributes that are not common between the two sets provide no predictability. Finally, the dataset is transformed into standard spreadsheet format by creating dummy variables for all categorical variables. The schema for the final dataset (before dummy variable transformation) can be found in Appendix A.

3 Predicting Propensity To Click

Five different machine learning algorithms are applied to the training set to fit a model to predict the probability of a click: logistic regression, decision tree with a max depth of two, decision tree with a max depth of five, linear discriminant, and naïve-bayes. To understand the effects of the stratification ratio, each of the algorithms are applied to different samples of the training data.

The performance of each algorithm is evaluated in a variety of ways. First, the accuracy for each algorithm is recorded. Because clicks are so rare, this is not the best performance indicator as models that predict more no clicks will be favored. Next, the precision and recall for each model is recorded. The precision and recall at different stratified samples are shown in Figures 1-2. As the percentage of no click impressions increases in the stratified sample, precision increases because the model predicts less clicks. Similarly, the recall decreases as the percentage of no click impressions increase in the sample. Logistic regression seems to be the best performing algorithm in terms of recall and precision because the rate of increase in precision is greater than the rate of decrease in recall at larger sample sizes.



Figures 1-2

Lastly, the algorithms are evaluated by a '90% Cumulative' measure. This measure sorts all impressions by their propensity to click, and finds percent of total impressions needed to generate 90% of the clicks in the dataset. Therefore, lower percentages are ideal. Figure 3 shows the 90% Cumulative measure for each of the algorithms.

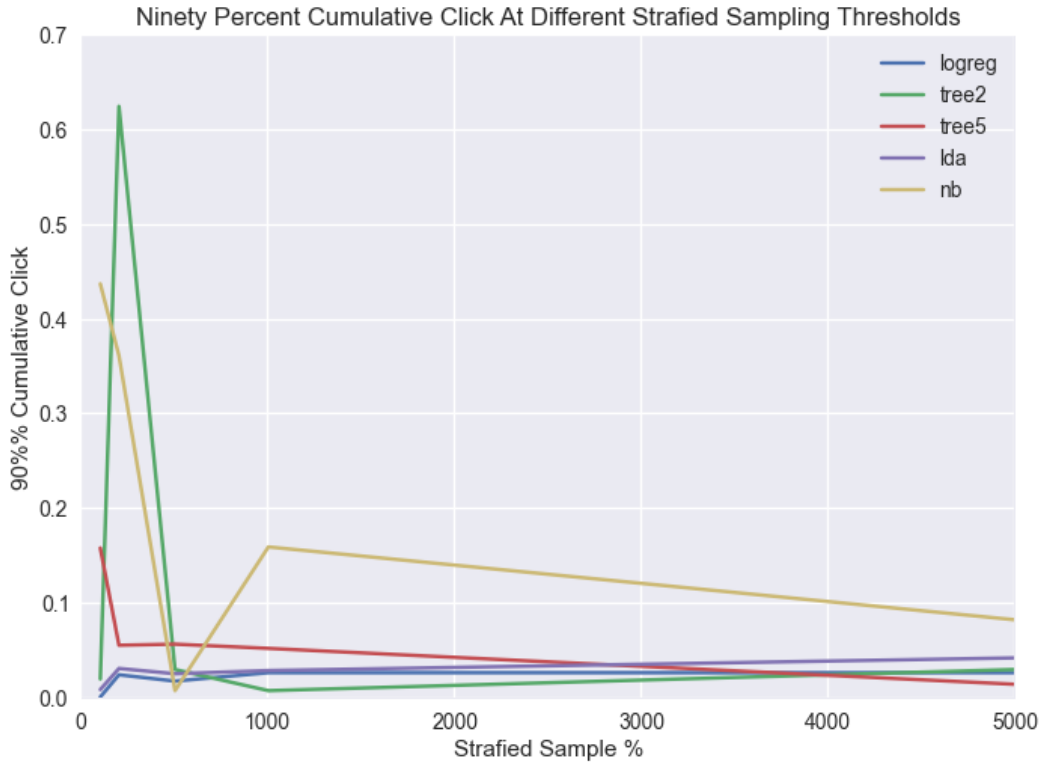


Figure 3

As shown in the figure, Naïve-Bayes and Decision Tree Classification (max depth of 2) at a low sample size are the worst performing algorithms. Logistic regression is the best performing model at lower stratified sample sizes and continues to be a good performer at increased sample sizes. The ten attributes that have the largest coefficients in the logistic regression model are shown in Figure 4. The model shows that ad size and whether the user is on a mobile device (android and iphone) are important predictors to clicks. Logistic regression is chosen as the model for all bidding strategy simulations in Section 4.

Attribute	Logisitic Regression Coefficient
adslotsize_120x600	1.501
android	1.434
adslotsize_360x300	1.289
adslot_5	1.248
iphone	1.015
linux	0.972
adslot_1	-0.863
region_394	-0.904
adslotsize_960x90	-1.052
adslotsize_200x200	-1.345

Figure 4

4 Bidding Strategy

A basic and common approach that is used to determine bids in a real time bidding (RTB) campaign is to bid the product of the estimated propensity to click and a constant that represents the target cost per click (CPC). It is widely thought that this approach yields the greatest results in a market with no supply or demand constraints and limitations. It can be easily shown that this method of bidding is not optimal in minimizing a cost per click metric when an advertiser is faced with a limited budget.

In this section, the performance of the logistic model created in Section 3 is evaluated by simulating an ad campaign using the bidding strategy of bidding the propensity to click (as generated by the model) multiplied by the advertiser goal. This bidding strategy is referred to as *Goal Bidding*. The performance of this algorithm is compared with the performance of flat bidding (referred to as *Flat Bidding*) to measure the lift in advertiser ROI. Finally, a bidding strategy that dynamically changes the scaling factor throughout the campaign (referred to as *Dynamic Bidding)* is proposed. An ad campaign is simulated using this method and compared to the performance of the goal bidding strategy. Success for the Dynamic Bidding strategy is measured by how close performance can come to the optimal Goal Bidding strategy.

4.1 Flat Bidding

The flat bidding strategy simulation goes through each impression in the test set and bids a flat amount. Flat bids range from 5 Yen Cost Per Thousand Impressions (CPM) to 300 Yen CPM. A 5 Yen CPM represents the minimum winning price in the dataset. A 300 Yen CPM represents the maximum winning price in the dataset. The performance for different flat bids can be seen in Figure 5. Click through rate increases as the flat bid increases while cost per click stays fairly flat for all levels of flat bids. This shows that there is a natural trade off between performance and price for a given impression.

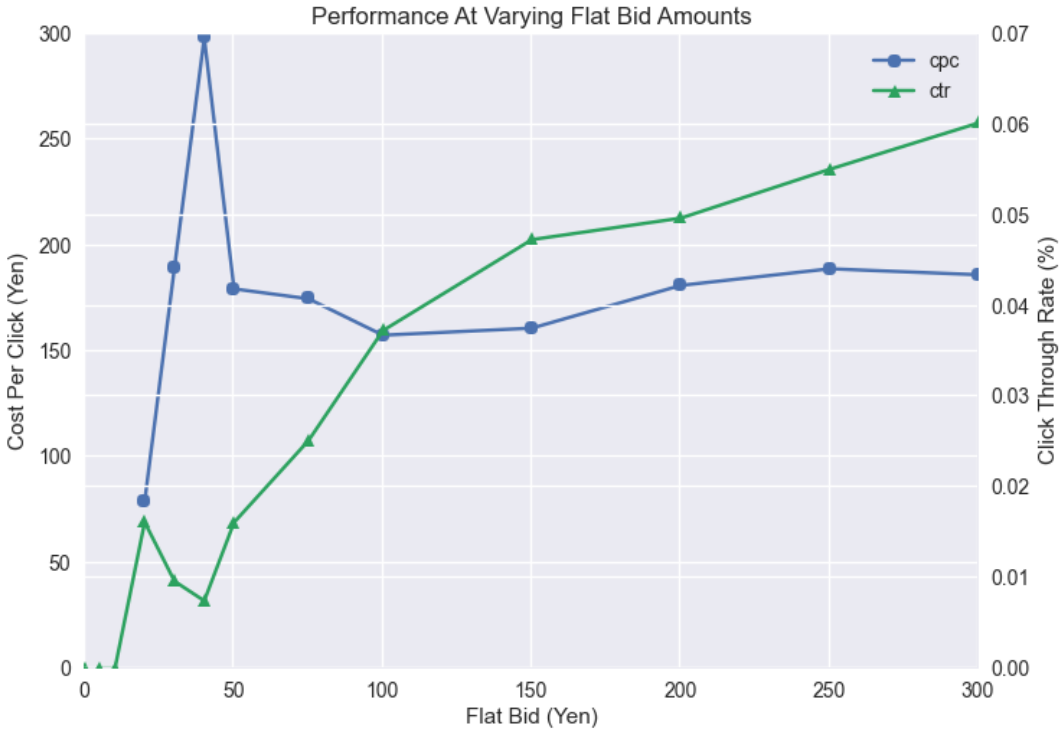


Figure 5

4.2 Flat Goal Bidding

Arbitrary CPC goals are picked to cover a range of bidding urgency, from being highly selective to nearly buying all impressions. In particular, the goals that are chosen range from 10 to 1,000. The performance for different goals can be seen in Figure 6. The click through rate increases quickly as the goal is increased. The quick spike in CTR indicates that the goal needs to be high enough to scale the bids high enough to buy expensive impressions that have a high propensity to click. The CTR then decreases for the highest goals, showing that the algorithm is doing a good job paying up for impressions with a high propensity to click. Cost per click gradually increases as the goal is increased, indicating that there is a tradeoff between ad spend and performance (higher goals backs into larger ad spend).

Figure 6

4.3 Dynamic Goal Bidding

It can easily be shown that a flat goal bidding strategy is inefficient for fixed budgets. For an actual campaign, if the goal is set too high, the campaign will overpace and deliver the budget before the campaign ends. If the goal is set too low, the campaign will underpace and the budget will not be delivered in full.

The dynamic goal bidding strategy dynamically changes the scaling factor (analogous to the goal in the flat goal bidding strategy) throughout the campaign. An initial value of 100 is used for the scaling factor, but is updated after the first 500 impressions are served. The scaling factor is updated by first calculating the spend urgency, as defined as the percent of the overall worth of future ad impressions needed to spend the budget. A random sample of 500 historical bids is then taken. The random sample is then sorted by the cost per predicted click in ascending order. The cumulative sum of the cost is computed, and the cost per predicted click for the impression that's cumulative cost is equal to the bid urgency is taken to be the scaling factor. This process is detailed in the following algorithm:

ALGORITHM: DYNAMIC BIDDING

- compute the total worth of future impressions in campaign
- calculate spend_urgency = % of worth of future the campaign needs to buy
- resample: randomly select 500 historical bids from the campaign
- compute cost per propensity for each impression in random set
- sort the random set by cost per propensity in descending order
- calculate the cumulative spend for the impressions
- return the cost per propensity for the first impression that has a cumulative spend greater than the spend urgency

repeat every 5,000 impressions

For a direct comparison, the budgets selected for the simulation are the same budgets that are spent for the flat goal bidding strategy. The performance for this model can be seen in Figure 7. Both CPC and CTR follow similar patterns to the flat bidding strategy. Again, CPC decreases for larger budgets, showing that the bidding strategy is successful in gaining utility for smaller demand.

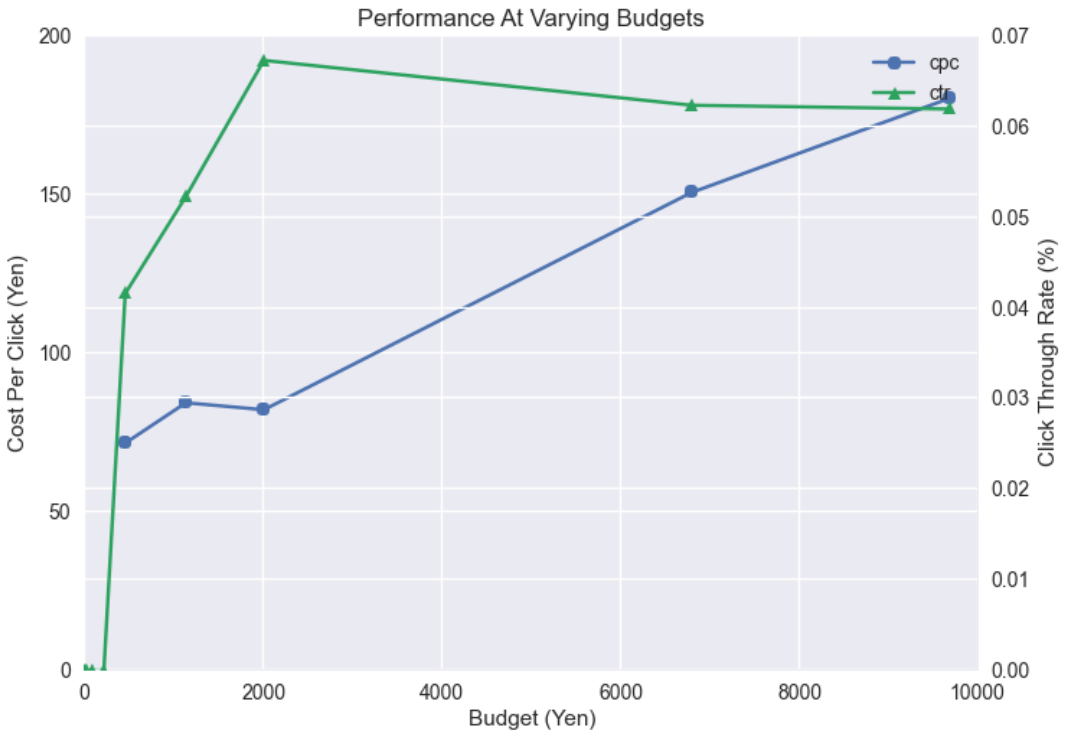


Figure 7

4.4 Comparison Of Methods

The three bidding strategies can be compared by looking at performance over a range of ad spend quantities. The ad spend for the flat bid strategy and the flat goal strategy were are not chosen but are decided by the flat bid or flat goal, but nonetheless, represent a nice range of ad spends. The dynamic goal budgets are chosen to be the same budgets that are spent in the flat goal simulation. The dynamic goal strategy did a good job at coming very close to spending the budget in full, but very marginal deviations from the budget did occur. This would not be a major constraint in real life as campaigns are ran over longer periods and the update in goal can be adjusted more frequently during the end of campaigns.

Figures 8-10 show the performance for the three bidding strategies over a range of budgets. CPC and CTR are both sub-optimal for the flat bidding strategy, showing that advertiser ROI is in fact increased by utilizing a bidding algorithm. The CPM is relatively consistent for all bidding algorithms, showing that the algorithm does a good job at identifying good performing impressions rather than just bidding low to achieve performance.

Figure 8

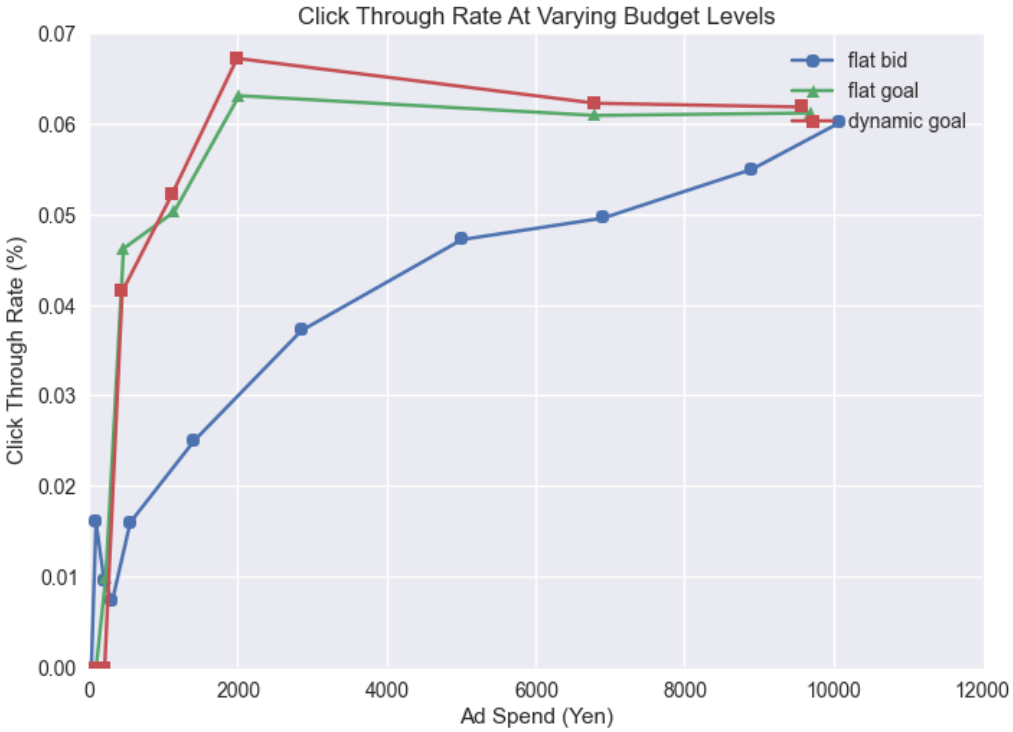


Figure 9

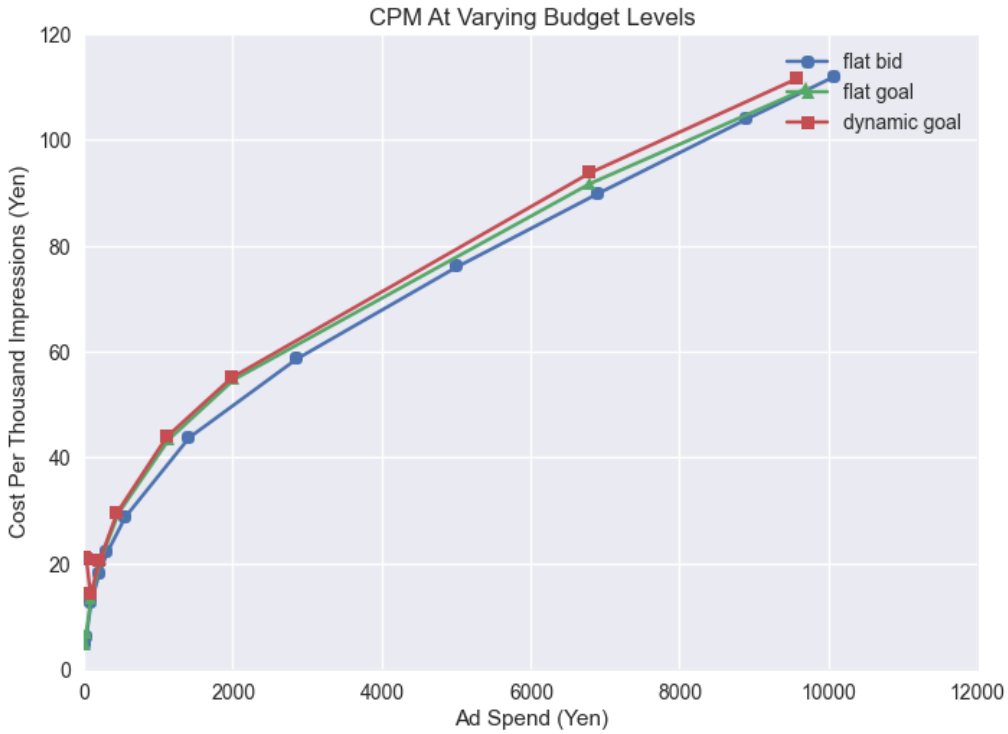


Figure 10

The lift of each bidding strategy against the flat bid strategy is evaluated at different budgets by comparing the CPC performance at similar budgets. This is shown in Table 1. The lift is highest for smaller budgets and decreases for lower budgets. This makes sense, as there is less room to be selective at higher budget levels.

Ad Spend – Flat Bid	Ad Spend – Flat Goal	Ad Spend – Dynamic Goal	Flat Goal / Flat Bid CPC Lift	Dynmaic Bid / Flat Bid CPC Lift
189.66	209.78	197.65	90%	0%
538.82	449.16	429.41	280%	251%
1,339.50	1127.81	1101.45	202%	206%
6,887.79	6783.75	6775.40	120%	123%
8,888.27	9678.85	9572.17	125%	105%

Table 1

Conclusions

The contributions of this paper are three fold. First, it is shown that advertiser ROI is increased by utilizing the logistic regression-bidding algorithm outlined in Section 3. For smaller budgets the lift in advertiser ROI is up to 250%. Second, it is shown that advertisers should expect greater ROI for lower levels of budget. This conclusion shows that the RTB market is indeed a buyers market as the utility gained by the advertiser increases with decreased demand or increased supply. Lastly, a real time bidding strategy is proposed that dynamically updates the scaling factor of the bids using a resampling approach. This dynamic bidding strategy mirrors the theoretical maximum performance of flat goal bidding with an unconstrained budget.

The conclusions shown in this paper are made using a handful of assumptions. First, the nature of the dataset only allows for information to be gained on impressions that are sold for less then 300 Yen. Further research would have to be done to ensure that the model can be extrapolated to cover higher priced impressions. Second, the remaining value of future impressions seen in the auction is calculated discretely when calculating the bid urgency in the Dynamic Bidding model. This technique is unrealistic in a real world setting, as this amount would not be known. It is not unrealistic to assume that a probabilistic model can be created to predict this amount with high accuracy, particular for campaigns that are in market for a long period of time. Lastly, the conclusions made in the paper are derived from just a sample of one day's of data.

Despite the mentioned limitations to the study, the conclusions made provide a solid framework for future work. First, additional models to generate propensity to click can be tested in the same manner as logistic regression is tested to find the best performing prediction model. Additional bidding strategies can also be tested in the framework. The dynamic goal can be fine tuned by changing the number of impressions that are resampled and the times that the goal is updated. Using more data over a larger amount of time should be able to yield an accurate model to predict the remaining worth of impressions left in the market. Lastly, testing the model on real world data against current models can be done to assess the overall bidding effectiveness

References

1. iPinYou Global Bidding: contest.ipinyou.com

Appendix A: Dataset Description

Attribute	Description
bid_id	Primary key – a unique identifier of each bid and impression resulting in a bid.
ipinyou_id	A unique identifier for each unique user seen in the data set.
timestamp	The time that the impression occurred
hour	The hour that the impression occurred
browser_chrome	Binary variable – one when the user uses a Chrome browser, zero otherwise
browser_ie	Binary variable – one when the user uses a IE browser, zero otherwise
browser_safari	Binary variable – one when the user uses a Safari browser, zero otherwise
browser_firefox	Binary variable – one when the user uses a Firefox browser, zero otherwise
mobile	Binary variable – one when the user agent identifies the user is on a mobile device, zero otherwise
iphone	Binary variable – one when the user agent identifies the user is on an iPhone device, zero otherwise
ipad	Binary variable – one when the user agent identifies the user is on an iPad device, zero otherwise
android	Binary variable – one when the user agent identifies the user is on an Android OS, zero otherwise
windows	Binary variable – one when the user agent identifies the user is on a Windows OS, zero otherwise
linux	Binary variable – one when the user agent identifies the user is on a Linux OS, zero otherwise
region_id	ID identifying each unique region
ad_exchange	Id identifying each unique ad exchange
domain	Hashed website domain of the impression
ad_slot_id	Unique ID to show the location where the ad impression will show up on the web page
ad_slot_size	The size of the advertisement slot
ad_slot_visibility	Represents whether the ad slot is above or below the fold
ad_slot	Fixed, popup window, background, float, or NA (not specified)
ad_slot_floor_price	The minimum price that the impression can be sold for
paying_price	The price paid for a given impression (second price reduced)

Appendix B: Python script

The script that runs all calculations for the analysis is in 'rtb.py'. To run, a folder named 'raw_data' containing the the click logs and impression logs for the test and train data (named click_test.txt, click_train.txt, imp_test.txt, imp_train.txt) must be placed in the same directory as the Python script.