Web Economics: Individual Report

Stylianos Rousoglou
Affiliate Student, Computer Science, University College London
stylianos.rousoglou.16@ucl.ac.uk

1. INTRODUCTION

Real-Time Bidding (RTB) is an increasingly popular approach to online advertising. For advertisers, the challenge in successfully automating ad auctions and bidding in real-time is the necessity to predict, with the highest possible accuracy, the likelihood that the ads displayed to a particular user will actually be of interest to them, so that a maximum number of impressions (which the advertiser pays for) have the potential of becoming conversions.

Given real-life ad impression data, a machine learning approach is often preferred in tackling this problem. In attempting to accurately predict user behavior, a thorough and educated study and use of the training data is required for useful results. This report will first present some data analytics that result from a basic exploration of the training and test data sets provided for the assignment. Some statistical information will then be presented for a general, high-level understanding of the data at hand. Subsequently, my personal approach to building a machine learning classifier and developing a bidding strategy will be detailed. The techniques employed throughout my individual strategy will be discussed, and different design decisions made will be defended. Finally, the results of the individual strategy will be presented and commented on using relevant evaluation metrics.

2. RELATED WORK

The extensive literature review performed covers a range of academic papers and articles. Work most relevant to this report includes *Real-time bidding benchmarking with ipinyou dataset* and *A logistic Regression Approach to Ad Click Prediction* (see References). The former is used as a model for the presentation of and the preliminary statistical analysis performed on the data, while the latter provides valuable insights into building a logistic regression classifier, which is the machine learning model used in my individual bidding algorithm, as well as into techniques for data pre-processing, data clensing and data reduction.

3. DATASET

3.1 Files

The dataset consists of three distinct data files, namely the training, validation, and test files. Machine learning classifiers are first trained using the training data set, *train.csv*. Subsequently, the *validation.csv* file is used to evaluate the performance of different classifiers and suggest which one may be the most accurate. While the two aforementioned files include user feedback information about each ad impression that is used to perform supervised learning and to evaluate the classifier respectively (in the case of RTB, user feedback refers to whether the ad was clicked or not, i.e. a binary variable), the third file, *test.csv*, is only used for testing the developed model, and thus does not contain the *click* data field.

3.2 Statistics

A quick look at the overall statistical results provides a high-level understanding of the nature of the data at hand. Analysing the data more closely will then allow for more interesting results, as patterns in the data will reveal that metrics such as the Click-Through Rate (CTR) of ad impressions vary greatly across attributes such as different advertisers or devices and operating systems.

num Imp	703,430
Clicks	539
Total Value	56,481
avg CTR	0.077%
avg CPM	80.30
avg eCPC	104.79

Table 1: Overall Statistics from *train.csv* (currency is Chinese fen)

Table 1 presents some overall statistics from the training data file; results on the validation file are quite similar in relative magnitudes, though *validation.csv* contains significantly less data.

At first glance, a few immediate observations can be made about the dataset. First, the number of impressions (num Imp) is orders of magnitude larger that the number of clicks. As a result, the average CTR is predictably small, precisely $CTR = 0.077\%$. Following this observation, the results for average Cost per Million Impressions (CPM) and average effective Cost per Click (eCPC) appear reasonable; given the scarcity of clicks among the thousands of impressions, an average cost of about 105 per *clicked* impression (avg eCPC) is justifiable.

The diversity of advertisers in the dataset lends itself well to a cross-advertiser analysis. Table 2 presents statistical results for the 9 distinct advertisers and allows for numerous conclusions to be drawn regarding the significance of the advertising agent in successful bidding. Firstly, all but one advertisers have a similarly low CTR, ranging from 0.018%

Table 2: Training Data across *Advertisers* (currency is Chinese fen)

Advertiser	Impressions	Clicks	Cost	CTR	CPM	eCPC
1458	140,312	128	9,694	0.091%	69.09	75.73
2259	38,466	7	3,580	0.018%	93.07	511.43
2261	31,459	10	2,809	0.032%	89.30	280.92
2821	60,416	34	5,382	0.056%	89.07	158.28
2997	14,239	67	900	0.47%	63.21	13.44
3358	79,522	67	7,354	0.084%	92.47	109.76
3386	130,193	94	10,006	0.072%	76.86	106.45
3427	118.577	90	9,601	0.076%	80.97	106.68
3476	90,246	42	7,156	0.047%	79.29	170.37

to 0.091%; for eight out of nine, $CTR < 0.1\%$. However, advertiser no.*2997* has $CTR = 0.47\%$, which is notably *more than 500% higher* than the CTR of the second most successful advertising agent, no.*1458*. It is then evident that, based on the dataset statistics, an ad from no.*2997* is *far more likely* to be clicked than an ad of any other advertiser. Expectedly, advertiser no.*2997* is an outlier in other calculated values as well: the $eCPC = 13.44$ is by far the lowest Cost per Click in the table, proving that given no.*2997* is by far the most successful in ad conversions. All advertisers have similar CPM values, and all but no.*1458* have reasonably comparable eCPC results.

The diversity of operating systems in the dataset also allows for certain conclusions to be drawn from a cross-OS analysis perspective. Table 3 presents the statistical findings when impressions are grouped by Operating System of the device used. An expected but nonetheless useful finding is the advantage that the *android* operating system seems to dominate over all other operating systems. The click-through rate for *android* devices is $CTR = 0.56\%$, twice the value of *linux* and *mac* devices and almost 1000% more than *windows* devices. One could speculatively attribute this disparity to the nature of mobile advertising, whereby full-screen ads on relatively small screens make it much easier for ads to be accidentally clicked, or even noticed. The *android* OS also has the lowest average CPM, with all other systems having similarly higher values, as well as the lowest eCPC, with *mac* and *linux* as runner-ups and *windows* costing about 1000% more per click.

4. APPROACH AND RESULTS

4.1 Data pre-processing

Before training the machine learning classifier, there is a significant amount of work to be done in order to prepare the data for "learning". First, a high-level analysis (presented in detail in section 3.2) is useful in deciding on the good data features to be used for classification, i.e. the ones that do correlate with differences in the Click-Through Rate (such as Advertiser and Operating System.) Accordingly, several data fields that either have unique or highly differentiated values, such as the *bidid* and the *userid*, will not be useful for classification and thus have to be removed. Thus, **data selection** is performed to remove those fields, as well as *IP* (unique to each user), *logtype* (1 in this dataset), *domain*, *url*, *urlid*, *creative*, *keypage*, etc.

Given the large volume of data with negative user feedback (click = 0) and the scarcity of impressions that led to conversions (click = 1), **undersampling** was used to increase the ratio of clicked to non-clicked impressions in the training data. The small bias towards clicked impressions in training helped correct the bias in the original dataset, where only 0.077% of all impressions were clicked.

4.2 Approach

Several regression and classification machine learning algorithms were considered by my team. The problem lends itself well for supervised learning; the training dataset does contain auction results as well as user feedback. It may be tempting to try *Linear Regression*, but there is a fundamental problem with that: the output of the linear model *can be less than 0 or greater than 1*, rendering the output meaningless. To solve this problem, *Logistic Regression* was introduced, with a mathematical model that actually restricts the dependent variable value in the range $[0, 1]$. In reality, the model is estimating *the probability* of a categorical outcome, i.e. the likelihood of one discrete output result versus another. The general Logistic Regression formula for two outcomes (0 or 1) can be written as

$$P(Y = 1) = \frac{1}{1 + e^{-}(a_0 + \sum a_i x_i)}$$

Though the independent variables may either be continuous or discrete, the result of the model is a probability of the dependent output taking the given discrete value.[1]

Though logistic Regression was not the approach finally undertaken by the team, it was the one i experimented with and tried to optimize in my personal solution. Logistic regression is a *binary classification* algorithm, which means that the result of its predictions is discrete by nature. This caters perfectly to our problem, namely deciding whether a user will click (click = 1) an ad or not (no click = 0). Also, all impression data used for learning is categorical, which is acceptable since predictor variables in Logistic Regression dependent variables may be both continuous or categorical. The underlying mechanism of Logistic Regression is Maximum Likelihood Estimation (MLE), a widely used statistical model for estimating the values of parameter variables that maximize the likelihood that the output takes the observed value.

The approach to solving the problem can be summarized as follows. First, the feature file *train.csv* is loaded into memory. Data selection and undersampling are performed as discussed in Section 4.1, and features that shouldn't be used for learning are removed from the data structure as

[1]https://www.quora.com/What-is-logistic-regression

Table 3: Training Data across *Operating Systems* (currency is Chinese fen)

OS	Impressions	Clicks	Cost	CTR	CPM	eCPC
android	18,361	103	1,299	0.561%	70.72	12.61
ios	573	0	10	-	88.13	-
linux	439	1	38	0.228%	85.79	37.66
mac	13,734	40	1,119	0.291%	81.44	27.964
windows	670,204	395	53,966	0.059%	80.52	136.62
other	573	0	50	-	86.83	-

well. Subsequently, feature extraction is performed: the feature-value mappings chosen for learning are tranformed into sparse matrices, to perform a technique known as *one-hot encoding*. One-hot encoding creates a vector for every possible string value any feature can possibly take. This of course wouldn't be possible should the data not be categorical. One-hot encoding tranforms complex feature-value mappings into simple (though very large) matrices with only binary digits, a conversion which is very helpful for mathematical computations as well as more successful machine learning classification results.

After one-hot encoding, the data is ready to be learned on. The selected feature vectors are passed into the *scikit-learn*[2] Python library, and a logistic regression model is built to produce CTR predictions. Since the dataset is very large, this step is the most time-consuming so far. After learning is completed, the validation dataset is loaded and some data manipulation is performed, as described above. The model is then used to calculate CTR predictions for every impression in the validation dataset and store them in an in-memory array. The real-time bidding simulation can then be performed, and its results are evaluated.

Although I experimented with various non-linear models, I attempted to optimize the linear model strategy, since the other members of my team worked on ORTB. Therefore, this is the solution I will be discussing. In this strategy, the bidding price for a given impression is proportional to its predicted CTR. Specifically, the formula can be written as $bid = C * (pCTR/avgCTR)$, where C is the *base bid* constant. Conceptually, the base bid would be the bid price for an impression with average CTR, i.e. for which the ratio $pCTR/avgCTR$ is 1. For impressions with higher-than-average predicted CTR, the ratio would be over 1 and the bid price would be proportionally higher than the value of the base bid.

After calculating the mean CTR in the training dataset, I varied the base bid value in the range $[0, 400]$ and repeated the evaluation process to decide on a good base bid value. The results are detailed below.

4.3 Results and Evaluation

Figures 1 and 2 plot the metrics CTR and CPC respectively as a function of the base bid value. At this point, it's clear that no single evaluation metric reveals the optimal base bid. Deciding on a good constant is a trade-off between volume of clicks, click-through rate, and of course cost. Specifically, although the number of clicks increases as the base bid value does, the CTR ratio is decreasing, as is the CPC. The maximum CTR is observed for a base bid value of 4. However, it produces only 10 clicks, whereas base

[2]http://scikit-learn.org

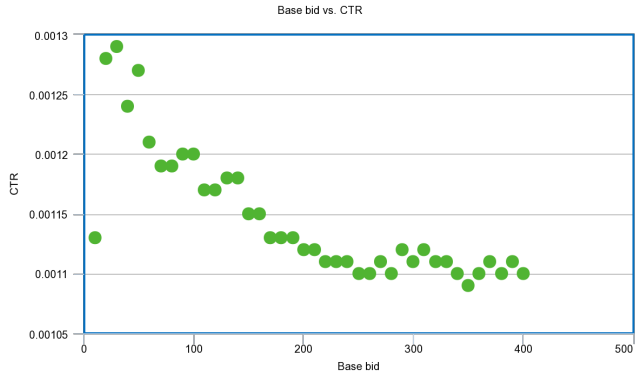

Figure 1: Base bid vs. CTR

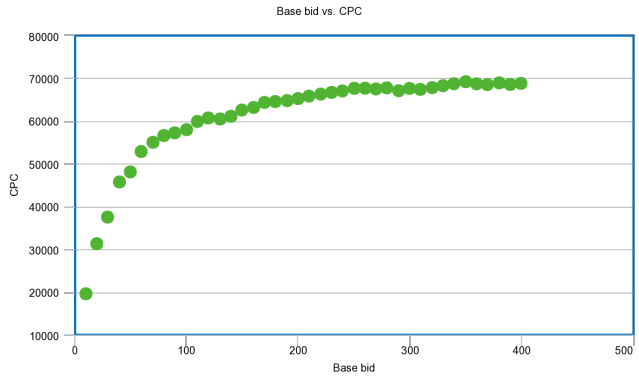

Figure 2: Base bid vs. CPC

bid values of 30, 60, and 150 produce 68, 113 and 149 conversions respectively, with CTR 0.13%, 0.12%, and 0.11% respectively. On the flip side, the CPC for the latter three values is 37.5, 52.9, and 62.7 Fen respectively. Therefore, selecting an optimal bid is subject to the goals of the bidding strategy. If solely optimizing CTR, one should pick a low base bid value, optimally 4. If optimizing number of clicks, on the other hand, the highest base bid value of 400 yielded the most, namely 176.

5. CONCLUSION

Our group met a few times before April to work and discuss the project, which became harder after we left London for Easter Break. Our initial meetings lay the groundwork for the project; we discussed the approaches we would undertake, wrote preliminary code, and came up with a timeframe that works for everyone. Michael wrote the bulk of

the code, being the one more comfortable in Python, with continuous input from Devin and myself, who tried to improve the algorithms, catch mistakes, and make the code more efficient. Devin worked on both the linear and non-linear models, whereas I optimized the constant and random bid strategies, wrote most of the group report, ran the code trials, and produced the graphs.

6. REFERENCES

[1] Xuehua Shen, Jun Wang, Shuai Yuan, and Weinan Zhang. Real-time bidding benchmarking with ipinyou dataset. *arXiv preprint arXiv:1407.7073, 2014.*

[2] Gouthami Kondakindi, Vinit Parakh, Sai Kaushik Ponnekanti, Aswin Rajkumar, and Satakshi Rana. A Logistic Regression Approach to Ad Click Prediction. *Mach Learn Class Project, 2014.*