Real Time Bidding Project Individual Report

Michael Whitman
University College London
168-182 Drummond Street
London, United Kingdom
michael.whitman.16@ucl.ac.uk

1. INTRODUCTION

This project deals with the real world problem of effective online advertising. The goal was to create a strategy that, using a large set of data from past advertising impressions, effectively predicts which ads will be clicked on and then makes an attempt at bidding and winning these advertisements, while losing the ones that do not get clicked. To do this, I created a strategy which uses machine learning on the dataset to get predictions on the probability that an ad will be clicked, and then applies a formula to generate a bid prediction. This strategy met modest success, and will be further detailed through this report.

2. OUTSIDE RESEARCH

Aside from the other documents referenced in this report, a few other sources were consulted in preliminary research. The scikit learn documentation was invaluable in becoming familiar with my chosen implementation library. A great deal of deliberation was put into deciding which model to use to get pCTR. This question helped give insight that led to the usage of the Random Forest Classifier into various models experimented with.

3. DATA EXPLORATION

The data given is segmented into three different files: train, validation, and test. The validation and test set both consist of 299749 impressions. The test set is used in the final evaluation of the model's ability to predict prices and purchase click producing ads, and therefore does not contain metrics such as whether an impression will yield a click or the price of the ad. The validation set is similar in use, except for it is used for testing prior to submission. Therefore, it contains these metrics not found in the test set, but they should only be accessed when evaluating a model's success. The train set is much larger than the other two, containing 2697738 impressions. It is used as it is named, for training the model before it is evaluated. It therefore contains all available fields. Inspiration for this section is lightly drawn from the provided source [1]

3.1 Basic Statistics

The training set contains 2037 impressions that yielded clicks, which is a very small portion of the overall number of impressions, giving a CTR of just 0.000754. The validation set has 226 clicks, and an identical CTR of 0.000754. These low CTRs indicate a very imbalanced dataset, which will require a balancing technique to be applied prior to or during learning.

The total cost of all impressions and all clicks for the training set is hardly relevant. However, it is relevant when looked at for the validation set. The total cost of all impressions in the validation set, calculated by summing their payprice attributes, which is what would be paid in the event of a bid being won, is 24,045.218. The total cost of just impressions that produce clicks is 24.511. This makes the budget an ample size to purchase desired ads.

The average cost for all bids in the training set, the average CPM, is 80.251. For the validation set, the average CPM is similar at 80.2178. The expected cost per click, or eCPC, of the training set is 106.6819, and similarly the eCPC of the validation set is 108.4558. The eCPC is higher, indicating that in general clicks are indeed more expensive than nonclicks, due to the higher value of the impression.

3.2 Field Analysis

The data provided the training and validation sets contain 26 fields, whereas the test data has 23 fields. The three fields not present in the test data are click, payprice, and bidprice. Bidprice is what the real highest bid was for an impression, payprice is what was paid, and click is whether or not the impression was clicked. Payprice is used in the evaluation stage, to determine whether or not a bid was won, and click is helpful in both evaluating the model's success and in the training phase, for helping determine what features lead to more clicked impressions. Click is the only real metric provided which can be construed to the buyer as a conversion, or a successful advertisement, so it is the most important and sole feedback. the other two aforementioned metrics provide information on others are willing to pay for the impression, although they cannot be used as inputs to any model, as they are not present in the test set. These two costs, as well as slotprice, are listed in CPM format, meaning they must be divided by 1000 to convert to CNY fen.

The remaining fields are all present in the test data, and therefore can potentially be used as or modified into features in a learning model. However, not all of these fields are useful, as they contain identifiers or nonsense values not relevant to the problem at hand. Other fields are more useful when modified, such as when decomposed or combined to form other features. Categorical data also needs to be considered for one-hot encoding, in order to be used for machine learning processes. Table 1 contains information about these fields, if it should be used, and if relevant how it should be used, and whether one-hot encoding should be used.

Models intending to predict bids will have to determine

Table 1: Usage of Data Fields

Field Name	Use/Usage	One-hot encode?
weekday	Use, can modify to determine if it is a peak bidding day	Yes
hour	Use, can modify to determine if it is a peak bidding hour	Yes
bidid	Don't use, is used for generating the test output	-
logtype	Don't use	-
userid	Don't use	-
useragent	Use, can decompose into operating system and browser used	Yes
IP	Don't use	-
region	Use	Yes
city	Use	Yes
adexchange	Use	Yes
domain	Don't use	-
url	Don't use	-
urlid	Don't use	-
slotid	Don't use	-
slotwidth	Use, can combine with slotheight to be area	Can encode, but not required
slotheight	Use, can combine with slotwidth to be area	Can encode, but not required
slotvisibility	Use	Yes
slotformat	Use	Yes
slotprice	Use	No
creative	Use	Yes
keypage	Use	Yes
advertiser	Use	Yes
usertag	Don't use	-

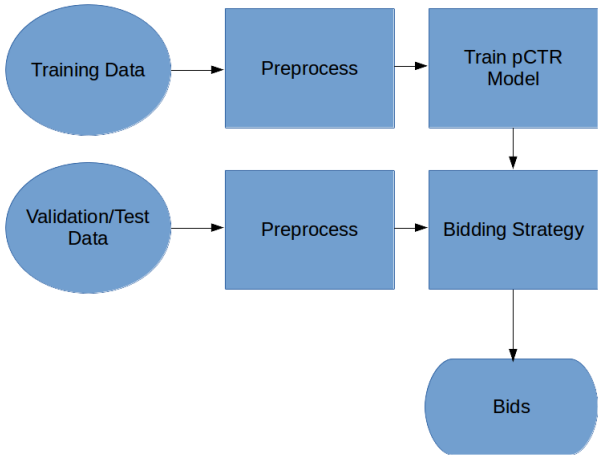

Figure 1: High-level process for predicting bids

how to transform these fields into features that can be used for learning. Because there are so many fields that, especially after encoding, becomes a rather large feature space, determining a method of feature selection is of some importance for both the speed and accuracy of the learning model.

4. INDIVIDUAL BIDDING STRATEGY

The strategy described here is the result of a great deal of experimentation and trial and error. There is certainly room for improvement, but it provided a good introduction into the problem and insight into the development of the full group model.

4.1 Overview

The process that my strategy undergoes can be seen in Figure 1. Two sets of data are loaded in, the training data and either the validation or the test set, depending on whether the model is being tested for improvement or to generate results for project evaluation. Each dataset must be run through a preprocessing phase, before it can either be used for learning or be evaluated. The learned model generates pCTR estimates, which are the core of how the estimation strategy generates its bid predictions. These are then reviewed, generating relevant statistics if the validation set is in use.

4.2 Preprocessing

The preprocessing phase is what is used to turn the raw data into a set of usable features to be input into the model, either during the training phase to tune the pCTR model or the evaluation phase to have the model generate results.

The first step is to attempt to balance the data. Because there are so many impressions that are nonclicks compared to clicks, there needs to be a way to better even this out. My solution was to undersample, which means that a hefty portion of the nonclicks are eliminated from the data being used by random selection. The amount of nonclicks kept is determined by a tuning parameter, which when multiplied by the number of clicks yields the number of nonclicks that will be kept. Currently, this parameter is set to 5 as this was what seemed to work well, but this can be easily changed.

The next step is to synthesize features where necessary. I modified the useragent for each impression by splitting at the underscore, making fields for the browser and operating system used. I combine the slot width and height into one value of area to simplify the dataset slightly. Instead of including the day and hour, I created features which are 1 if it is a peak hour or a peak day, which would imply greater chances of a click. Finally, based on the OS information retrieved from useragent, I determine if a mobile device is being used and put that in a feature.

Next, a one-hot encoding process is applied. This turns categorical fields, which almost all of the fields can be treated as, are converted into multiple fields, with values of 1 or 0 depending on what the impression's original field value was.

This immensely expands the number of fields, all of which can now be treated as features. However, this feature set must be decreased in order to only keep statistically relevant features that will best tune the pCTR model. In order to do that, I apply a k-best selector, which selects k number of features which are determined the most statistically relevant by a chi squared test to whether or not an impression is a click. Currently, k is set to select 100 features from the expanded feature set.

After all of this, the features are ready to be input into a learning model.

4.3 Learning the pCTR Model

The most important component for this real-time bidding strategy is having a reasonably accurate way of getting the pCTR for an impression, or in other words determining how likely it is that an impression will yield a click. Since the most important metric for the project is the click-through rate, impressions yielding clicks are highly valued.

To do this, I used a logistic regression to make predictions on the click likelihood of an impression. This takes in the features that were extracted from the training data, as well as the click data, to attempt to determine how certain features can predict a click. I experimented with other models as well, such as a random forest classifier, but they did not seem to offer any improvement over the logistic regression.

This model's default behavior was to output its prediction of whether any input impression would be a click. However, to extract the pCTR values, I had it output the underlying probabilities of these predictions, which can then be used as pCTR values for bid prediction.

4.4 Bid Prediction

The pCTRs obtained must be transformed according to a bidding strategy in order to attempt to win bids resulting in clicks, and lose bids that do not.

The equation that I use is the ORTB1 formula, which was derived from the cited paper [2]. It is of the following form:

$$\sqrt{\frac{c\theta}{\lambda} + c^2} - c \tag{1}$$

λ is the Lagrangian multiplier, c is a constant, and θ is the input pCTR value. Through trial and error, the best settings for the tunable values that I determined for my configuration were $\lambda = 8.33 * 10^{-5}$ and $c = 45$. These settings seem to yield good bid predictions, but could likely be improved with more tweaking.

4.5 Evaluation

Table 2: Results

Statistic	Value
Click-Through Rate	0.000740
Conversions	217
Spend	22267
Average CPM	75.975
Average CPC	102.6173

Table 3: pCTR for Clicks and Nonclicks

Statistic	Value
Average Click pCTR	0.181478
Average Nonclick pCTR	0.163576
Median Click pCTR	0.151066
Median Nonclick pCTR	0.151111

To evaluate the success rates of the strategy, a testing data set must be fed in, which can be either the validation or test set. Data from one of these is loaded in, preprocessed into features as described before, and used to predict pCTR for each impression. This is passed into the pCTR model, outputting bids for use. If the test set is in use, these results are then output to a file for submission.

If the validation file is being used, the bids predicted are compared to the payprice field to determine the successes and failures of the strategy. Relevant statistics, such as the CTR and Average CPC, are then output for analysis.

5. RESULTS

I ran my strategy on the validation set with the given budget constraint of 25000 CNY fen. Statistics from the run are recorded in Table 2.

5.1 Analysis

My strategy performed fairly well, winning the lion's share of the clicks, while having a click through rate much better than that of the entire dataset, by a factor of 10. I ended with money still left in the budget. In addition, the average CPM and average CPC were also lower than that of the whole dataset, though not by much.

6. CONCLUSION

6.1 Pitfalls

Although this strategy appears to post middling results, it is hard to tell if they reach a high enough standard. However, there is definitely room for improvement. What I interpret as the biggest flaw of the whole system is the pCTR prediction system, which does not in my opinion yield sufficiently predictive results to be improved much by any changes to the strategy bid predictor. Table 3 demonstrates some statistics about the pCTR for all clicked impressions vs. nonclicked impressions in the validation dataset when run through the pCTR model. It can be clearly seen that although the averages provide some differentiation, the median for click pCTR is in fact lower than that for nonclick pCTR. This indicates a need to improve pCTR prediction in order to substantially improve the strategy.

While running the program, I also noticed that many of the bids being placed on clicks were abnormally high. This

could indicate, given the lack of differentiation in the pCTR values, that too much is being spent on nonclicks, which would lead to too many of these being won and thus greatly reducing the CTR, and increasing average CPM and CPC. Efforts to reduce this overspending would work well when done in tandem with pCTR improvements.

6.2 Possible Improvements

There are many possible improvements that could be made to the pCTR model. Different models other than logistic regression could be applied, and tuning parameters could be modified to attempt some improvement. I have done some work with a random forest classifier, and with some work I believe this may at some point be a better pCTR predictor.

More potential improvement lies in the features. I have tried other methods of feature selection than k-best selection, such as by using a cross validated logistic regression as a means of selecting feature from a model, but I was unable to gain any meaningful improvements. I have also performed experiments in further feature pool expansion by attempting to generate features based on their interrelatedness, but was unable to get this to work in a meaningful way.

There are other improvements that could potentially be made to the bidding equation, such as by selecting a different model, such as ORTB2, which if given time to optimize constants could potentially be a better strategy.

Overall, the model seems to perform fairly well, especially at finding and winning cheaper bids. With some improvements to the pCTR prediction, it should do very well when incorporated into the group strategy.

6.3 Contributions

For this group project, my main focus was on writing the code for the strategy and implementing the strategies while simultaneously researching new methods. Devin focused on analysing our implementation and determining constants, providing some help on research and giving ideas. Stelios wrote most of the group report and provided some ideas through the process.

6.4 Github

Code that I used for my individual report can be found here: https://github.com/mibewh/Compm041Project. It is under the individual folder, in the file michael_whitman_strategy.py

7. REFERENCES

[1] W. Zhang, S. Yuan, and J. Wang. Optimal real-time bidding for display advertising. *Real-time bidding benchmarking with ipinyou*, 2014.

[2] W. Zhang, S. Yuan, J. Wang, and X. Shen. Real-time bidding benchmarking with ipinyou dataset. *Proceedings of the 20th ACM SIGKDD conference on Knowledge discovery and data mining*, pages 1077–1086, 2014.