### POLITECNICO DI TORINO Department of Control and Computer Engineering (DAUIN)

Master's Thesis

# Cost per Click and Clicks Prediction in Autobidding System



Academic Supervisor Silvia Chiusano Candidate Giulio Collura

Industrial Supervisor Vishnu Purushothaman Sreenivasan

Academic Year 2017/2018

#### Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains less than 65,000 words including appendices, bibliography, footnotes, tables and equations and has less than 150 figures.

Giulio Collura April 13, 2018 

#### Abstract

Online advertisement is the main source of revenue for most online companies. What is crucial in this market is not only the quality of ads or targeting the right users, but being competitive on prices for click or action for every interaction with the users.

In our scenario, advertisers set a monthly budget to be spent in ads. The objective is to fully utilize the available budget, and at the same time maximize the value in terms of click conversion rate.

The purpose of this work is to improve predictions for click and cost per click for new potential advertisers whether they decide to go with a higher or lower budget.

Displaying a more accurate prediction on our subscription page would lead at increasing the number of advertisement contracts, better trust from customers, and increased long-term value.

An exploration of different possible models has been carried on in order to create a linear relation between budget and cost-per-click for each advertiser. In the end, a single model capable of predicting discrete click values at specific budget values has been chosen, and it is ready to be deployed in production.

This model presented better offline performances than the previous model, with on average twenty percent improved performance.

In parallel with this research, a small research on the new autobidding system that would guarantee a faster reaction to our advertisement landscape changes. This system will simulate real-time updates. In order to support faster bid value changes, a work on the system infrastructure has been studied and deployed. Dedicated to my family and friends who support and encourage me everyday.

# Contents

1	Intr	oduction 9											
	1.1	Problem											
	1.2	Contributions											
	1.3	Structure of this work											
<b>2</b>	Bac	kground 13											
	2.1	Ads on Yelp											
	2.2	Ad Autobidding System											
		2.2.1 Terminology											
	2.3	Industry examples 16											
3	Mo	del for cost-per-click or click prediction 19											
	3.1	Proposed evaluation for the final model											
	3.2	Use cases											
	3.3	Evaluation methods											
<b>4</b>	Data analysis 23												
	4.1	Business features											
	4.2	Spend curves											
		4.2.1 Sampling spend curves to generate additional training data 26											
	4.3	Budget to Cost-per-click relationship											
<b>5</b>	Pro	posing and Testing Various Models 33											
	5.1	Status quo model											
		5.1.1 Random Forest bid bootstrap model											
		5.1.2 XGBoost Joint bid bootstrap model											
	5.2	Linear regression-based model											
		5.2.1 Model design											
		5.2.2 Evaluation											
	5.3	XGBoost-based models											
		5.3.1 Slope and intercept											

#### CONTENTS

		5.3.2	Slope and pCPC	37
		5.3.3	Model design	38
		5.3.4	Evaluation	38
	5.4	Neural	Network-based model	39
		5.4.1	Model design	40
		5.4.2	Evaluation	40
	5.5	Click p	prediction model with XGBoost	41
6	Rea	ltime 4	Autobidding	45
7	Con	clusior	15	49
	7.1	Future	works	49

# Chapter 1

# Introduction

Today, online web advertisement is the main source of revenue of many companies, such as Google, Facebook, and Yelp.

It is crucial, in order to survive the competitiveness of the market, to have efficient and automated systems capable of showing the best add to the users.

#### 1.1 Problem

Businesses have access to a Yelp for Business Owner portal, that features several tools to allow owners to keep their page up-to-date and enroll with Yelp to start advertising on other Yelp pages.

Among the steps for setting up Yelp Ads, the new potential advertisers can choose how much they are willing to spend daily on average (see fig. 3.1) for setting up Yelp Ads. From a quick look they can see what is going to be their predicted average monthly cost-per-click and a predicted number of clicks. Both of these predicted values are independent of what is the budget chosen, in fact, the number of clicks grows linearly with the budget.

In reality, both of these assumptions are not true and wildly inaccurate. In this work, correlation and causality of budget to cost-per-click will be investigated, in order to find the best way to model different budget values or ranges for the same businesses to a different value for their cost-per-click or number of clicks.

Over a month, cost-per-click and number of clicks are strictly tied through the following relation:

$$cost-per-click = \frac{budget}{\#clicks}$$
(1.1)

Due to the above equation, prediction of one of the two are basically the same, but the accuracy and the error metrics to evaluate the goodness of our predictions may be different.

Our goal is to have customized prediction for every budget value or range, in order to get more trustworthy values showed to users, and improve the conversion of new advertisers, which means increase the number of new subscribers, and in the long term, their retention.

#### **1.2** Contributions

The whole project broke up on several steps and iterations. The main goal was a better model capable of predicting different values of *cost-per-click* or #*clicks* on different budget inputs.

The first step was a deep data analysis of the relation upon budget and cpc per different category. Since for a single advertiser we did not have more than one chosen budget, an investigation was brought on to increase the dataset with more than one budget to cost-per-click relations for each business. Then, several attempts in modeling the problem were built and tested with different metrics, to see whether the predictions were accurate or better than the previous existing model, which just predicted one single value. The most critical part was establishing a correct and fair metric that would take into account the newly generated data the new model was trained on.

After several trials and errors, both in data collection of new samples, model building and evaluation, a new model, in many ways similar from the previous one, has been chosen. This new model can predict different values for the number of clicks a new business may potentially receive on average in their first month with a given budget.

#### **1.3** Structure of this work

In the next chapter a background introduction on what is autobidding in Yelp and the main concept in its system will be explained in details.

Following up the final model of this project will be presented with its characteristics, goals and use cases. In the same chapter, the evaluation methods will be presented, in particular, why they have been chosen and how they will be used.

In Chapter 4, a deep data analysis will be presented to the reader, also how and why it has been decided to include in the training data more samples of budget to clicks/cost-per-clicks for each business, and where these new datapoints come from.

Next up, in Chapter 5, all the models that have been built and tested will be presented and explained, with all their different characteristics and why they have been discarded or chosen.

Not strictly related to this experiment, but related to this internship experience, an introduction to Realtime Autobidding will be presented in Chapter 6. Realtime Autobidding is the new challenge of our team in providing updated bids for our advertisers on a shorter than a day time period, in order to optimize fulfillment and reaction to traffic changes, and eventually to decrease the average cost-per-click and increase retention of advertisers.

As conclusion, in Chapter 7, a retrospection on what could have been done differently, in terms of model building approach and solution to the problem. Next potential steps for this project will be presented, including how an online experimentation will be considered and evaluated.

### Chapter 2

# Background

#### 2.1 Ads on Yelp

Advertisement on Yelp provides to business owners a powerful tool to increase their visibility. In Figure 2.1, it is possible to see an example of search page, in this case *Home Cleaning in San Francisco*. The two displayed ads are from businesses in the area the user is searching for. Ads results look about the same of search results, except for the tag Ad in blue on the left corner, and they feature reviews count, average rating, location and small review.

What businesses should compete in the auction are selected with ad targeting algorithms based on user location, business rating, location, and category.

The auction is then run with the selected businesses and in this case, the highest two bidders are selected.

The best review, not just the one with the highest rating, but the one that better describe the business is selected by ad creative, as well as the overall appearance is chosen.

If a user clicks on one of the two ads, he will be redirected to the business' page, and charged of the amount the bidder he won against bid, according to the second price auction mechanism (Edelman et al., 2007).

#### 2.2 Ad Autobidding System

Every advertiser, when decides to start investing in online advertisement, sets an amount of budget that he is willing to spend every month.

Autobidding is basically an optimization problem for cost-per-click campaigns. The optimization goal is to help cost-per-click campaigns to achieve maximum clicks with a fixed budget.

In the ads team, we always try to balance between our users, our advertisers and Yelp revenue. In other words, optimize for high click conversion rate (CCR) for users, lower cost per click (CPC) for advertisers and better fulfillment for Yelp itself. These metrics are good



Figure 2.1: Example of Yelp search result with advertised businesses.

proxies for user retention, advertiser ROI and customer lifetime value. We can see that Autobidding wants to achieve fulfillment goal without hurting CPC and CCR goal.

For every ad slots, an auction is run with several candidates using generalized secondprice auction (GSP, Edelman et al. (2007)). Generalized second-price auction is a auction mechanism. Each bidder places a bid. The highest bidder gets the first slot, the secondhighest, the second slot and so on, but the highest bidder pays the price bid by the second-highest bidder, the second-highest pays the price bid by the third-highest, and so on. The last bidder, in our scenario, will pay a fixed price, which established a priori.

#### 2.2.1 Terminology

In this subsection, we will explain the basic terminology and concepts related to Autobidding.

Advertiser An advertiser is generally a Yelp business that wants to be showed and get noticed in users' search or on other businesses' pages. They can decide to start a new campaign any time of the year.

**Campaign** A campaign is an advertiser's instance in the bidding system. Campaign have a monthly budget, that can have a billing period spanning from the first of a month to its end, or from any arbitrary day and last thirty days; the latter are called rolling month budgeted campaigns. A single business can have multiple campaigns at once, for multiple location businesses, each campaign would compete in different geographical areas.

**Impression** When an advertiser's ad gets showed to a user, this is called impression. Some years back, this was the only way to charge advertiser for the service provided by ads companies, but nowadays the industry has switching to billing for click or action.

**Bid and auctions** Every advertisement spot on the website and mobile apps is assigned through an auction between several candidates eligible to get shown on that space. The eligibility is evaluated by ad targeting algorithms, but the final assignment is managed through an auction system. The auction compares the best bids set from the advertisers, through autobidding, and selects the winner. What the advertiser is going to pay for each click is the second winning bid. Bids are set upon several factors, mainly expected cost per click and remaining budget for the spend period.

**Cost per mile (CPM)** Each time the advertiser wins an auction he will be shown to the user. The advertiser will then pay an amount, according to the auction, to be delivered to the user, no matter if it will get clicked or not.

**Cost per click (CPC)** Every time an user clicks on any ad from an advertiser, the latter gets charged by an amount of money that is, in Yelp scenario, the second winning bid in the auction.

**Spend curve** Spend curve is a prediction of given a bid for a certain advertiser what would be the predicted spend and number of clicks at the end of the month, if the bid stayed the same through the whole month. Bids are set daily to get a total spend slightly higher the available budget, then cap the spend at the total budget, in order not to incur in overspending.

**Full-served campaign** Full-served campaigns are subscription model campaigns that are sold to businesses by Sales reps over the phone and they are contracts of different, predefined length. Each advertiser can choose one of the monthly budget suggested by Yelp. These are the most common type of campaigns, and their billing period – period of time the campaign is active – generally spans over a calendar month.

**Self-served campaign** Self-served campaigns are generally initiated by the business owner himself from his Business Owner application, with or without the help of a Sales rep. The predefined budgets are different from Full-served campaigns and the contract can be interrupted any time by the advertiser. The billing period is generally over a rolling month and starts the day the business becomes an advertiser.

#### 2.3 Industry examples

In this section we will compare our system with other online advertiser platforms, such as Google (Google AdWords Help, 2018) and Facebook (Facebook Help Center, 2018). They both use automated bid auctions in order to maximize the creation of value in terms of discovery and clicks for advertisers, while still providing relevant new contents to users.

Automated bidding takes the charge of tuning the bid price for each bidder in order to meet the requirements requested by the advertiser, may this be expressed in a bid strategy, in the case of Google.

Due to the less complexity of our system, Yelp does not offer any bidding strategy different from the default that maximizes clicks for advertisers.

**Google** Google offers both automatic and manual bidding. The latter allows the advertisers to pick their own bid price for each auction.

Many different automated bid strategies are offered by Google, from click maximization, conversion pacing and maximization, to search outranking. Conversion pacing sets a target cost per acquisition (CPA), and gets as many conversions as possible, while some conversions may cost more or less than the target.

#### 2.3. INDUSTRY EXAMPLES

**Facebook** The same auction system, second-price auctions, is used at Facebook to determine what ad should be displayed to a user. Same as Google, different bid strategies can be chosen by the advertisers, given the different possible nature of advertisement shown to users. Total value of a bid takes into account several factors, such as max bid set, ad quality and relevance.

### Chapter 3

# Model for cost-per-click or click prediction

The cost per click (CPC) of an advertiser is a complex function of several factors, some being the business' rating, the budget, and category.

Our goal is to give the new advertisers a predicted CPC for the budget they are setting for the first time on our platform. For these advertisers, we do not have historical data, or previous campaigns expenses and fulfillments.

The most straightforward idea is to have a model able to represent a function f:

$$CPC = f(budget, biz\_feat) \tag{3.1}$$

This function f could be any prediction model, able to output increasing CPC given an increase in budget.

Another path that will be considered is predicting for a certain business the number of clicks it may get during a month. Number of clicks follows this relation:

$$\#clicks = \frac{budget}{pCPC} \tag{3.2}$$

While the evaluation of this two proposed models will be pretty similar, how the accuracy will be evaluated will differ: we assume that from a user's perspective, a 10-20% inaccuracy on cost per click prediction, would be perceived differently if the same error is on number of clicks received. Number of clicks could depend on several factors, like traffic and users' interests.

To simplify our problem, traffic and seasonal patterns will not be considered in building our model, since we want our prediction to be an average for every month, not just their first month of advertisement.

#### **3.1** Proposed evaluation for the final model

The evaluation of the new candidate model is crucial for our work.

The most intuitive idea would be comparing the predicted cost per click with the average cost per click charged for each advertiser during a particular month. There are a couple of issues with this approach: every advertiser generally keeps the budget unchanged across several months, and similar advertisers may have the same cost per click, independently from their budget.

This two issues make an interesting challenge for this project. Since we cannot test our performance on real data, we have to research a metric that would take into account what is the increment of cost-per-click or number of click on an increment of budget, both from a qualitative and a quantitative point of view.

On the beginning on the project we focused on looking mostly at qualitative results, like what was the slope of budget to cost-per-click variation, because we were looking on learning the full spend curve around the real budget the campaign had. Later on, since this approach was difficult to understand and did not bring the expected result, we switched to a simpler approach, that consisted in learning the samples from the spend curve on fixed budget values.

As seen in Figure 3.1, the budget shown on the webpage are \$300.00, \$450.00 and \$750.00. Additionally to these three budget values, we will sample also at other most common budget values, such as \$150.00 (minimum allowed budget), \$275.00, \$475.00, and \$725.00.

#### 3.2 Use cases

In Figure 3.1, the general use case for such model would be proposing better clicks or cost-per-click prediction depending on which daily or monthly budget the user selects. Currently, the shown number of clicks is calculated through Equation (3.2).

Another use case would be *bid bootstrapping*. This means that for new advertisers for which we do not have any historical data, given by previous participation in auctions, we predict a mean cost-per-click, and predict their spend curves.

#### 3.3 Evaluation methods

We will base our final evaluation on how closest is the new model on the spend curve samples, independently from the real budget they had, and also look at those self-served campaigns that chose one of the predefined budget on their first and second month.

The two metrics used are root mean square error (RMSE, eq. (3.3)) and proportional

#### 3.3. EVALUATION METHODS



Figure 3.1: Business owners backend page that helps the new potential advertiser choose the budget they want to invest in online advertisement.

mean absolute error (PMAE, eq. (3.4)),.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_{real} - y_{pred})^2}$$
(3.3)

$$PMAE = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_{real} - y_{pred}|}{y_{real}}$$
(3.4)

RMSE is differentiable, thanks to this property, we used this also as optimization function in building our models. But since it is biased towards higher values, we decided to evaluate on the PMAE metric, which guarantees to always be close within a certain percentage towards the real value. 22

## Chapter 4

# Data analysis

In this chapter, we are exploring and analyzing our dataset, in order to highlight the pattern and the main characteristics.

#### 4.1 Business features

Each business is described by several features, that identify its categories and how it may differ from other business or advertiser. Below, all the features are described.

- average\_bid: average bid for all cpc impressions delivered on opportunities a business is eligible to compete in, which means business eligibility on an opportunity based on geoquad and category criteria
- rating: autobidding takes the rating column from the business table in yelp-main mysql clusters.
- open\_ratio: given by the following relation:

$$open\_ratio = \frac{open\_opportunities}{expected\_impressions}$$
 (4.1)

$$expected\_impressions = \frac{total\_budget}{average\_bid \cdot \frac{total\_impressions}{total\_clicks}}$$
(4.2)

on the opportunities a business is eligible to compete in. the expected impressions value is how many time advertiser's ads will be displayed to the users.

• average\_candidates: average number of candidates that made it to the auction on opportunities a business is eligible to compete in. A candidate is an advertiser that participate in an auction for an opportunity.

• opportunities\_per\_impression: given by the following relation:

$$opportunities\_per\_impression = \frac{open\_opportunities + competed\_opportunities}{expected\_impressions}$$

$$expected\_impressions = \frac{total\_budget}{average\_bid \cdot \frac{total\_impressions}{total\_clicks}}$$
(4.4)

an open opportunity has at least one empty slot. a competed opportunity has no open slots, but delivered cpc impressions.

- review\_count: number of reviews for the business
- max\_bid: maximum bid for all cost-per-click impressions delivered on opportunities a business is eligible to compete in
- competed\_ratio: given by the following relation:

$$competed\_ratio = \frac{competed\_opportunities}{open\_opportunities + competed\_opportunities}$$
(4.5)

an open opportunity has at least one empty slot. a competed opportunity has no open slots, but delivered cpc impressions. calculated over opportunities a business is eligible to compete in

• inventory\_str: given by:

$$inventory\_str = \frac{total\_impressions}{total\_page\_opportunities}$$
(4.6)

since there is typically more than an opportunity slots per page opportunity, this can be greater than one. calculated over opportunities a business is eligible to compete in.

• inventory\_ctr: given by:

$$inventory\_ctr = \frac{total\_clicks}{total\_impressions}$$
(4.7)

calculated over opportunities a business is eligible to compete in.

- categories are stored as one-hot vector: all boolean indicator features that are true if the category is in the set of the business's explicit categories or the ancestors of the explicit categories according to the yelp category tree.
- global\_cpc\_cpc\_mean, stdev: mean and stdev cost-per-click of globally similar advertisers. advertiser similarity is based on the explicit categories of the business.

#### 4.1. BUSINESS FEATURES

- trust\_rating\_avg: unweighted average rating for a business. this is similar to the rating showed on yelp website.
- global\_cpc\_budget\_mean,stdev: mean and stdev budget for globally similar cpc advertisers.
- log\_exact\_available\_opportunities: log transform of open slot opportunities where a business is eligible to compete and the opportunity's category is an exact match with one of the business's explicit categories.
- log\_num\_search\_appearances\_per\_day: log transform of the number of times the business appeared in search.
- local\_cpc\_cpc\_mean,stdev: mean and stdev cpc of locally similar cpc advertisers
- current\_program\_age: number of months the business has been advertising. this would be 0 at prediction time.
- global\_num\_similar\_adv: raw count of globally similar advertisers
- max\_fee: budget of the business' campaign.
- global\_cpc\_fulfillment\_mean,stddev: mean and standard deviation of fulfillment of globally similar cpc advertisers
- local\_cpc\_budget\_mean: mean budget of locally similar advertisers
- log\_expand\_available\_opportunities: log transform of the open slot opportunities a business is eligible to compete in
- global\_cpc\_ctr\_mean: mean CTR (click-through-ratio) of globally similar CPC advertisers
- local\_cpc\_ctr\_mean: mean CTR of locally similar advertisers

In our feature set, the features related to a business' budget are open\_ratio (eq. 4.1), opportunities\_per\_impressions (eq. 4.3) and max\_fee. During prediction time with the previous model, these three features were not available and not used. Only in training they were utilized to generate the model.

In our new model we eliminated these three features, because we wanted to only have one input feature called *budget* that, upon change, would change the final prediction of the cost-per-click or number of clicks.

#### 4.2 Spend curves

For setting bids daily we are interested in finding for each campaign what is the winning bid price in each auction. Winning bid price is the minimum bid value that campaign should have set in order to win that particular auction.

We collect a few set of statistics about what is achievable if we set a campaign's bid at a particular bid level on historic auctions. Increasing bids will get higher number of impressions and a larger amount of spend. All these values are collected in a table called *bid optimizer table*. The tuples can be plot in a graph and the result is called *spend curve*, fig. 4.1.

In Autobidding, spend curves are calculated for each advertiser to determine their bids in a month. Spend curves are predicted to calculate how many opportunities and how much an advertiser will spend, given a bid value. In other words, if an advertiser will bid every day a certain bid, he will very likely get impressions and spend some amount of his budget through the month.

Since we know with a pretty good accuracy how likely a user will click on an ad, which is the *predicted click-through-ratio* (pCTR), we can multiply this value with the predicted impressions and obtains the predicted number of clicks.

Spend curves are generated daily given historical data and by looking at all active campaigns, every month are different since each month the advertisers may be different.

#### 4.2.1 Sampling spend curves to generate additional training data

If we consider the relation spend to bid and bid to  $pCTR \times impressions$  (#clicks), we can easily construct a curve and bind together total spend and number of clicks.

Now, if we want the total spend to be equal to the advertiser's budget we could obtain the number of clicks the advertiser would get if he had that total spend as budget.



Figure 4.1: Example of spend curve for a campaign in a month

Thank to this new relation, we can sample more datapoints for every advertiser in the

form of pairs (*budget*, #*clicks*).

#### 4.3 Budget to Cost-per-click relationship

In this section, a short analysis on budget and cost-per-click features for all advertisers will described, in order to make sense of the differences between each root category.

Root categories in Yelp are, by definition, categories without a parent category. The category tree is a directed acyclic graph, with a single category being child of multiple parent category. The root categories are:

- Active Life (active);
- Arts & Entertainment (arts);
- Automotive (auto);
- Beauty & Spas (beautysvc);
- Education (education);
- Event Planning & Services (eventservices);
- Financial Services (financial services);
- Food (food);
- Health & Medical (health);
- Home Services (homeservices);
- Hotels & Travel (hotelstravel);
- Local Flavor (localflavor);
- Local Services (localservices);
- Mass Media (massmedia);
- Nightlife (nightlife);
- Pets (pets);
- Professional Services (professional);
- Public Services & Government (publicservicesgovt);
- Religious Organizations (religiousorgs);

- Restaurants (restaurants);
- Shopping (shopping);

The main categories we are mostly interested in, in terms of potential incoming revenue, are: Home Services, Local Services, Health & Medical, and Restaurants.

Overall, the total budget of advertisers is distributed according to Figure 4.3. In this plot, we have taken out the campaigns with budget higher than \$1000, since they constitute a small portion of our dataset. In Table 4.1, there is the list of the most common budgets in our training dataset.

max_fee	$\operatorname{count}$	%campaign
27500.0	26384	34.9
47500.0	11630	15.3
15000.0	5911	7.8
30000.0	5407	7.1
72500.0	4135	5.4
97500.0	2610	3.4
20000.0	2124	2.8
45000.0	2069	2.7

Table 4.1: Most common budgets

In Figure 4.2, cost-per-click is expressed in USD, and its distribution resembles a power law distribution, with the most common values under five dollars.

Budgets, on the other hand do not follow any particular distribution, from table 4.1, it is possible to see how the most common budgets – \$275 and \$475 – are coming from full served campaigns section 2.2.1, because self-served campaigns are less common in Yelp systems at the moment.

Even though our model purpose will be primarily to make predictions for self-served campaigns, we will consider both full-served and self-served campaigns, this way more data will be available and it will be easier to compare this our new model with the existing one.

Number of clicks histogram, fig. 4.4, follows a similar distribution as the power law seen for cost-per-click. Very few campaigns get more the five hundred clicks over a month, and most of them get few clicks a day.

Considering the root categories, the average budget, fig. 4.5, varies in each one. In our scenario, we are mostly interested in looking at homeservices, food and restaurants. The former category generally prefers having higher budgets, while the latter tends to have an average lower budget, but since they a higher amount of clicks, fig. 4.7, they have lower average cost-per-click. The trend is reversed for homeservices. In this category, since businesses may get more value out of a click/new client, for example a plumber can bill clients up to thousand of dollars for some special work, it is reasonable to have higher cost-per-click values, fig. 4.6.



Figure 4.2: Histogram of cost-per-click values of campaigns in training dataset, in USD.



Figure 4.3: Histogram of budget (max\_fee) for campaigns in training dataset, in USD cents.



Figure 4.4: Histogram of number of clicks for campaigns in training dataset.



Figure 4.5: Average campaign budget in each root category



Figure 4.6: Average cost-per-click in each root category



Figure 4.7: Average number of clicks in each root category

### Chapter 5

# Proposing and Testing Various Models

In this chapter, the various iterations are proposed and explained. Different approaches have been taken, some of them were slight improvements over some base model, other models, like the neural network one explained below, were introduced as a fresh different approach.

First of all, the status quo, which is the model that is currently employed in production and predicts only one value for the cost-per-click or number of clicks. Status quo comprises of two different kinds of models, Random Forest and XGBoost models. Both take as input only business' features, and no matter what the budget is set to, the predicted value would always be the same; predicted number of clicks would follow the Equation (3.2).

After, different kinds of experimentations are proposed, like linear regression built on top of status quo, or predictions of different values that would be part of an equation to map business features and budget to cost-per-click.

Later on, a different test using a neural network built with Tensorflow has been put into practice to predict a single cost-per-click value with business feature, with an additional layer with budget input to predict the final value.

These models only produce a cost-per-click prediction. Due to project iterations, we will only evaluate this models and the subsequent iterations only on cost-per-click error to ground truth. Since all these experimentations turned out to be inconclusive, it has been decided to reformulate the problem to a different approach: predicting clicks. Errors on click predictions can be interpreted differently, we care about being very accurate on lower values, while we can be more forgiving on bigger click numbers. In these iterations we will look at PMAE as our reference metric, while retaining RMSE as learning optimization function, since it is differentiable.

#### 5.1 Status quo model

In this section, the *status quo* model will be introduced with its characteristics. *Status quo* stands for the current model active on production.

Although, there could be only one model active at a time on *status quo*, we consider both models because they are both the most advanced reference we can compare ourselves against to.

#### 5.1.1 Random Forest bid bootstrap model

The current model built as a Random Forest (Ho, 1995) that predicts cost-per-click takes as input a reduced amount of business feature, like rating, review count, one-hot encoded categories and inventory features (average/min/max bid value, available opportunities, etc).

It does not take as input budget, but it has some *budget related features*, like open\_ratio and opportunities\_per\_impression. Also, this model does not take as input features like global,local\_cpc\_cpc\_mean,stddev or max\_fee, and other features based on similarity with other advertisers.

This model performances are listed in Table 5.1. The values are referred to computing RMSE and PMAE on the cost-per-click value prediction against the real value of our dataset.

Table 5.1: RMSE and PMAE of RF bid bootstrap model
--

Dataset	RMSE	PMAE
Train set	361.091232108	0.282419102582
Test set	514.291354076	0.298904059972

#### 5.1.2 XGBoost Joint bid bootstrap model

This XGBoost (Chen and Guestrin, 2016) model is an improvement of the above mentioned Random Forest *status quo* model, but it takes as input more business related features, like similarity with other businesses, like mean and standard deviation of cost-per-click and budget, both on a local scale and global scale, as described in section 4.1, for this reason this model is named *joint*.

This model performances are listed in Table 5.2. The values are referred to computing RMSE and PMAE on the cost-per-click value prediction against the real value of our dataset.

Table 5.2: RMSE and PMAE of XGBoost bid bootstrap model

Dataset	RMSE	PMAE
Train set	237.894622864	0.170011977264
Test set	450.185779633	0.243018817437

#### 5.2 Linear regression-based model

We use the first XGBoost model to predict an initial value for the CPC, then we apply a linear regression model to catch the spend curve slope.

Through cross-validation on our test dataset, we identified a couple of best coefficients for the linear regression:  $w = [1.18820787e^4, 1.01515142e^0]$ .

The motivation behind this approach is to learn the slope of a simplified spend curve, Section 4.2.

We will also perform a visual evaluation of this model, because we are interested on seeing what is the predicted slope of our model when comparing on spend curves.

#### 5.2.1 Model design

The model design is described by Figure 5.1. It consists of predicting a first pCPC value using an existing bid bootstrap model, in our case we will be using the XGBoost-based one, excluding any budget related feature. Once we predicted a first cost-per-click, we put this value in another model, along with a budget value, and predict the final prediction. This way, we try to predict a linear relation between budget change to cost-per-click change.



Figure 5.1: Simple visualization of this model two-step structure.

#### 5.2.2 Evaluation

In Table 5.3, we computed the RMSE of our new model. From a quick observation, it is noticeable that including a linear regression model on top of our XGBoost model, does not introduce major improvements. If we look at a visual representation of our predictions on budget changes, we can see that this model fails to catch the slope in the spend curves, as seen in Figure 5.2b.



(a) On X axis, the real CPC for a certain business, on Y axis the predicted CPC from linear model (blue plot) and predicted CPC from XGBoost model (without budget features, red plot).



(b) Visual evaluation of some campaigns spend curves and this model produced slope

Figure 5.2: Linear regression-based model

RMSEJoint XGBoostXGBoost w/o budgetXGBoost w/o budget + RLTest450.185451.768447.637Train237.894185.102183.900

Table 5.3: Linear regression scores

#### 5.3 XGBoost-based models

In this section, we explore how to model the linear relations in the form of

$$CPC = \alpha \times budget + \beta \tag{5.1}$$

by using XGBoost to learn a mapping between business features and  $\alpha$  and  $\beta$ , that could be easily interpreted as slope and intercept coefficients.

With this kind of model, we attempt to build a simplified algebraic expression of spend curves, by only learning their linear section. We do not look at non-linear regions, because they may contain unexpected, either too low or too high, values.

#### 5.3.1 Slope and intercept

With XGBoost, we build two models to predict slope and intercept and use them in the following equation:

$$CPC = slope \times budget + intercept \tag{5.2}$$

The slope parameter can be obtained by sampling the spend curve around the campaign budget, max\_fee value, and performing a linear regression of first order. The line inclination gives us the slope coefficient.

The intercept can be sampled from the spend curve cutting the y-axis or the global budget mean, namely the global\_cpc\_budget\_mean column; in this case the equation needs to take into account the offset given by the budget mean:

$$CPC = slope \times (budget - budget_{global\_budget\_mean}) + intercept_{global\_budget\_mean}$$
(5.3)

#### 5.3.2 Slope and pCPC

Similarly to what we did in the previous model, but instead of predicting the intercept we predict the budget the campaign should have and we use the predicted CPC from the first XGBoost CPC prediction model:

$$CPC = slope \times (budget - budget_{predicted}) + CPC_{predicted}$$
(5.4)

This model may recall the above-mentioned Linear Regression-based model (section 5.2); this time, instead, we have instead of inputting pCPC and budget into a linear model, we

predict pCPC and  $budget_{predicted}$ , using two different XGBoost models. With another third model we learn the campaign *slope* coefficient from the spend curve, obtained as above.

 $budget_{predicted}$  is necessary because the pCPC will implicitly include budget in its prediction, and the linear equation has to be centered around that budget. Unfortunately, predicting budget for a business is not simple, since it is based just on business owners' preference.

#### 5.3.3 Model design

The design of our *slope* models is fairly simple, we independently train our coefficients in different XGBoost models, with the same input features, but different target labels. A first model will be trained against slope coefficients, another one against budget, and a third against the predicted cost-per-click, similarly like we did in the linear regression-based model (section 5.2).

In Figure 5.3, a logical representation of this model composed of three XGBoost models: pBudget prediction model, *slope* prediction model, and pCPC prediction model. We take these three values and we compute the equation for the given budget and compute the final *cost-per-click*.



Figure 5.3: Visual representation slope, budget and predicted cost-per-click

#### 5.3.4 Evaluation

We trained our models on August 2017 campaigns, and tested on September 2017 unseen campaigns, which are completely new advertisers without any historical data. Since there are campaigns that span several months, we exclude those, since they may have the same or similar cost-per-click from month to month.

Model	Train RMSE	Train PMAE
Bid Bootstrap	361.091232108	0.282419102582
Joint Bid Bootstrap	237.894622864	0.170011977264
Slope+Intercept Bootstrap	346.698892089	0.283462138426
Slope+Budget+CPC Bootstrap	318.829817025	0.253797951209

Table 5.4: Train RMSE and PMAE comparisons of *status\_quo* and the new models

Our new model training performance, as seen in Table 5.4 is on par with our *status* quo models, but when we tested in prediction of cost-per-click on unseen data, the slope and intercept model RMSE was two times higher, and almost three times higher on slope, budget and pCPC model than the Joint Bid Bootstrap model (section 5.1.2).

Table 5.5: Test (next month unseen data) RMSE and PMAE comparisons of *status\_quo* and the new models

Model	Test RMSE	Test PMAE
Bid Bootstrap	514.291354076	0.298904059972
Joint Bid Bootstrap	450.185779633	0.243018817437
Slope+Intercept Bootstrap	998.389404237	0.446846736893
Slope+Budget+CPC Bootstrap	1388.45969652	0.47486242685

In Figure 5.4, we display a visualization of predictions through the slope and intercept model. This model from a visual point of view looks more accurate than the simple regression model built on top of cost-per-click prediction from section 5.2, but its RMSE and PMAE are too far from being accurate. Also training multiple models at a time does not seem feasible on the long run. For these reasons, we will not invest more research in improving this model.

#### 5.4 Neural Network-based model

With the Neural Network we do not target the single parameters, but we try to model business features to three parameters that will be inserted in the equation:

$$CPC = \alpha \times (budget - \gamma) + \beta \tag{5.5}$$

And then back-propagate in order to minimize the error on multiple values for each business.

The motivation behind this model relies on having a single model to be trained, capable of performing the same operation of the slope, budget and pCPC model, in a single step, thus more efficiently.

We build this model using Tensorflow. This time instead of using an one-hot vector to represent the categories the advertiser belonged in, we opted for category embeddings.



Figure 5.4: Visual representation of predictions on varying budget for the slope and intercept model, compared to spend curve values.

#### 5.4.1 Model design

From a design point of view, this model may seem more complex than the previous ones. In this case the model we are training is single deep neural network built of three independent neural networks that get the same input; predict independently three values, *alpha*, *beta* and *gamma*; that are combined with the input budget in the equation above (eq. (5.5)) to predict the final *cost-per-click* value. In training mode, we compare the prediction to the true value, compute the square error and back-propagate to the three neural networks and update the weights.

Each neural network perform a linear regression to compute a single value that goes into the equation.

#### 5.4.2 Evaluation

In our tests, the best configuration we found through cross-validation was three hidden layers of 64 neurons with Relu activation function and no activation on the output layer (single neuron). The loss function is RMSE and optimizer is Adam (Kingma and Ba, 2014).

The neural network experimentation gave on one hand very promising results, because compared to slope and intercept XGBoost-based models seen in the previous section, this model performed just as good and it is simpler to manage. The main downside is that neural networks are difficult to tweak, and although we performed several tests to improve the



Figure 5.5: Diagram of neural network structure

Table 5.6: Neural network training RMSE

Model	RMSE	PMAE
Joint bid bootstrap	295.053612161	0.41990999296
Neural network	420.119192388	0.61812970368

above results, we did not conclude with a better model on a single cost-per-click prediction.

From visual evaluation of Figure 5.6, we notice that of neural network model gets somewhere close to slope and values of a campaign spend curve, but it is not accurate enough for our likings.

#### 5.5 Click prediction model with XGBoost

Since the research of a general model capable of predicting at almost any budget value a different cost-per-click was inconclusive and did not bring any major improvement while

Table 5.7: Neural network test RMSE

Model	RMSE	PMAE
Joint bid bootstrap	450.185779633	0.243018817437
Neural network	878.710166223	0.552720018125



Figure 5.6: In red, campaign spend curve budget to cost-per-click; in blue, neural network prediction for campaign given the budget; blue dot is joint bid bootstrap prediction.

comparing to the old models, we decided to change and simplify the scope of our problem with a different formulation.

We decided to sample and predict spend curve values that, for each advertiser, would link a specific budget value to a certain amount of clicks. The budget values were predefined as the most common budgets and those showed on the Biz Owner interface (fig. 3.1).

First of all, we sample the spend curve for each campaign at specific budget values, which means looking at the expected spend if the campaign uses a bid value in all auctions in a month. From the bid value we obtain the number of clicks that a campaign may achieve by the end of its billing period.

Instead of researching and building a linear equation capable of mapping any budget at a specific cost-per-click or number of clicks, we train an XGBoost model that would predict only number of clicks at a specific budget value.

To evaluate this model we will look at the performances in predicting number of clicks at \$300, \$450 and \$750.

To increase the PMAE score, which means getting more accurate on smaller values prediction at cost of losing a little of accuracy on larger values, we train our model using as training labels log(y).

From the above tables, we can see that this new model performs significantly better than the Random Forest model introduced in section 5.1.1.

To prove that our new model was consistent in getting better results, we considered

Table $5.8$ :	RMSE fo	or predictions	$\operatorname{at}$	budget	\$300	compared	$\operatorname{to}$	status	quo	Random	Forest
model											

Dataset	Click pred model	$\mathtt{status\_quo}\;(\mathrm{RF})$	Improvement
Training	40.35	56.48	28.56%
Test $(Sept)$	41.37	52.84	21.70%
Test real (Sept)	35.83	37.77	5.13%
Test $(Oct)$	41.13	53.65	23.33%
Test real (Oct)	44.48	55.00	19.11%

Table 5.9: PMAE for predictions at budget \$300 compared to  $status \; quo \; {\rm Random \; Forest model}$ 

Dataset	Click pred model	status_quo	Improvement
Training	0.18	0.25	26.39%
Test $(Sept)$	0.25	0.28	12.21%
Test real (Sept)	0.27	0.31	12.57%
Test $(Oct)$	0.25	0.29	14.71%
Test real (Oct)	0.28	0.34	16.25%

Table 5.10: RMSE for predictions at budget \$450 compared to status quo Random Forest model

Dataset	Click pred model	status_quo	Improvement
Training	48.81	63.34	22.94%
Test $(Sept)$	50.26	59.30	15.24%
Test real (Sept)	50.18	48.79	-2.85%
Test $(Oct)$	51.84	63.02	17.74%
Test real (Oct)	50.80	59.87	15.14%

Table 5.11: PMAE for predictions at budget \$450 compared to  $status \; quo \; {\rm Random \; Forest model}$ 

Dataset	Click pred model	status_quo	Improvement
Training	0.17	0.27	35.72%
Test $(Sept)$	0.24	0.31	21.69%
Test real (Sept)	0.25	0.30	16.96%
Test $(Oct)$	0.24	0.30	21.81%
Test real (Oct)	0.26	0.33	20.05%

Dataset	Click pred model	status_quo	Improvement
Training	55.93	89.21	37.30%
Test (Sept)	64.11	83.93	23.61%
Test real (Sept)	52.65	78.20	32.67%
Test $(Oct)$	65.38	84.56	22.68%
Test real (Oct)	55.06	81.88	32.75%

Table 5.12: RMSE for predictions at budget \$750 compared to  $status \; quo \; {\rm Random \; Forest model}$ 

Table 5.13: PMAE for predictions at budget \$750 compared to  $status \; quo \; {\rm Random \; Forest model}$ 

Dataset	Click pred model	status_quo	Improvement
Training	0.16	0.33	51.58%
Test (Sept)	0.23	0.39	39.76%
Test real (Sept)	0.20	0.29	28.03%
Test $(Oct)$	0.23	0.35	33.81%
Test real (Oct)	0.27	0.31	13.15%

another month as additional test data, and we noticed that the percentage improvement was consistent. Additionally, since all train and test data contains points sampled out of spend curves, we wanted to look at campaigns that really had a specific budget value during a month: we introduced *test real datasets*. These datasets only contain unseen campaigns that had \$300, \$450, or \$750 budget.

The improvement on *real* campaigns is less significant because their bids were set with the model we are comparing ourselves against. For this reason there is a sort of bias towards *status quo*: the predicted number of clicks is independent from the selected budget and the initial bid was set starting from *status quo* prediction.

## Chapter 6

# **Realtime Autobidding**

One of the goals of autobidding is to be adaptive to changes in the system. The advertising allocation and pricing can keep changing based on several factors:

- Changes in advertising budget. Eg. addition of new campaigns, removal of old campaigns, change in budget of a campaign;
- Traffic fluctuations based on how much users are browsing or searching a particular category.

In our system, if any of these events happen during the day, we do not apply any modification to existing bids until the next day, when values are recalculated; until so, our system has an incorrect estimate of share for each campaigns as it's completely based on previous history.

A solution we could attempt would be to change bids or pricing for each campaign more frequently so that all campaigns gets adjusted to the modified competition landscape.

Another aspect we are interested in is finding the right bid value that would guarantee fulfillment of a campaign, in other words, we seek to find convergence to the desired bid value. By updating bids once a day, thirty times over a month, we might never get close to the ideal bid for each campaign.

There are two major goals for realtime bidding:

- Adaptiveness to changes in competition and traffic
- Achieving convergence

As regards of adaptiveness, we have models capable of predicting major changes in traffic, such as yearly seasonal patterns, like increase or drop of traffic on a particular category. Knowing this, the system may respond early by settings bids either high or low based on the estimation for the future. Whereas a completely reactive system, would wait until the errors happen and then correct for the errors.

Another part of the problem is attaining convergence at the current level of competition and traffic. Each campaign would need to change their bid based on the recent view of competition landscape and only after several iterations of this we might get closer to equilibrium.

Given these difficulties in modelling the system a reactive controller (eg. PID, proportional-integral-derivative, controller) would be able to achieve better results in achieving convergence. Using a reactive control would also mean we adapt to changes in competition or traffic drops automatically. This approach has been tested by other online advertisement system (Zhang et al., 2016), with great results.

In Figure 6.1, it is displayed how bid updates work currently for a campaign. The existing pipeline is not well optimized as this usually happens for all campaigns only once a day.



Figure 6.1: Autobidding system scheme. ad\_landscape is the Autobidding service that computes for all campaigns new bids. Kew and SNS are queues. y-m stands for yelp-main, the main component which Yelp relies on. adindexer is another service, in charge of storing campaign and bids information.

We can see that there are lot of intermediary components before the bid gets updated into each of the above mentioned tables. Of these the most slow part is *update\_ad\_store\_worker*. It would be a better idea to bypass this worker by providing *update\_bid* endpoint in *adindexer*. Our goal is to update bids every fifteen minutes. We added an endpoint in Yelp-main to push real-time bid updates on SNS. SNS is Simple Notification System and it is service offered by Amazon Web Services. It collects simple JSON messages that can get processed by downstream consumer, which, in our scenario, is adindexer.

The real-time message update is composed by the new bid value and the campaign identifier. All campaign's bids will be updated with the new bid value, in yelp-main database, in *AdBidHistory* table and the ElasticSearch index. The latter is what interests us the most since it is queried every time an auction is run.

We cannot simply push the bid updates to ElasticSearch because we need to collect and update information on the main database, and we cannot just use the old pipeline because it would require loading additional data that does not change every fifteen minutes.

The new pipeline is capable to update a single bid, by the time it is pushed onto the first KEW by real time batch, in about a minute. In Table 6.1, some examples of latency of a single bid update in our new pipeline.

In Figure 6.2, an example of total working time to update all the campaigns in our system with a new bid value. The total time also refers to pushing all the campaign updates on the first Kew by the realtime worker. As it can be noticed, to update all our campaigns less than 5 minutes are required. Considering not all campaigns' bids may be updated every time interval, this is a good throughput for our experimentation to start with.

trace_id	latency	times
918ed24d7c5a0a7a	68.259583	03/09/2018 14:04:49.508005
		03/09/2018 14:04:54.401509
		03/09/2018 14:05:21.006964
		03/09/2018 14:05:57.767588
d1c2cc8641842e05	68.244046	03/09/2018 14:04:49.515024
		03/09/2018 14:04:54.395999
		03/09/2018 14:05:20.992559
		03/09/2018 14:05:57.75907
e6c1e41f88a8d754	68.243207	03/09/2018 14:04:49.507864
		03/09/2018 14:04:54.390327
		03/09/2018 14:05:20.985094
		03/09/2018 14:05:57.751071

Table 6.1: Timestamp examples for bid updates



Figure 6.2: Total update time from bid update generation in ad\_landscape to the final write in ElasticSearch database

# Chapter 7

# Conclusions

Identifying and learning a relationship between different budgets to cost-per-click for campaigns in autobidding systems is not easy nor straightforward. In second-price auction systems, the final cost-per-click an advertiser will end up paying does not really correlate to its total monthly budget, rather to the remaining available spend. Whereas bids are set with budget fulfillment as optimization goal, a high bids may not be challenged by equally high bids.

We also faced the issue of not having enough data at our hands. Using spend curves was a necessity if we wanted to infer any kind of causality in the budget to cost-per-click or clicks relationship, and simply looking at our data was not enough to be able to build any model that would guarantee different cost-per-click prediction based on budget. At the same time, recreating the spend curve was too difficult, given the non linearity of the relation, and any possible approximation would have resulted in accumulating errors and in the end bad predictions.

We had to reformulate the problem, thus focusing our research in predicting an average click counts per new advertiser, based on the historical data of other and similar advertisers. This time, instead of forcing a linear relation between budget and clicks, we just learn at some predefined budget values the amount of clicks an advertiser would be able to get.

This new model allowed us to have the same level of accuracy as the previous model on the same metrics and a better accuracy in predicting for different budget values, with on average twenty percent improvement.

#### 7.1 Future works

The next step for the click and cost-per-click prediction will be to launch an online experiment by selecting businesses that will have the next interface with more accurate predictions. The success of such experiment will be evaluated on advertiser conversion, which is how many businesses start an advertising campaign, starting budget, whether a higher or lower budget get picked, and on long term value, which is how less likely are advertisers to churn.

Ideally, if new potential advertisers are seeing different clicks predictions for each budget they can select, they would feel more confident about the value they are getting out of their investment. With customized number of clicks under each displayed budget, they would be lead to choose the budget that would potentially guarantee them the best value. If our predictions are accurate, and advertisers do get the amount of clicks they expected, they would be more satisfied with our advertisement program, and more likely to stick with Yelp in the subsequent months.

On real-time autobidding system, the maths for the new bid value calculation will be researched, since it will contain a feedback loop based on the changes happened since the last update, simply recalculating the bids as we are doing daily will not guarantee convergence.

After a first controller has been written and tested offline, we will perform an online testing limited to a set of advertisers in a particular area, while retaining the old system for everyone else. In case the experiment is stable, we will gradually deploy the new system to a larger user base, until it becomes the only way of updating bids for advertisers.

# Bibliography

- Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. CoRR, abs/1603.02754, 2016. URL http://arxiv.org/abs/1603.02754.
- Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242-259, March 2007. URL http://www.aeaweb.org/articles? id=10.1257/aer.97.1.242.
- Facebook Help Center. How do the facebook ad auctions work?, 2018. URL https://www.facebook.com/business/help/430291176997542.
- Google AdWords Help. About automated bidding, 2018. URL https://support.google.com/adwords/answer/2979071?hl=en-GB.
- Tin Kam Ho. Random decision forests. In Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1, ICDAR '95, pages 278-, Washington, DC, USA, 1995. IEEE Computer Society. ISBN 0-8186-7128-9. URL http://dl.acm.org/citation.cfm?id=844379.844681.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.
- Weinan Zhang, Yifei Rong, Jun Wang, Tianchi Zhu, and Xiaofan Wang. Feedback control of real-time display advertising. CoRR, abs/1603.01055, 2016. URL http://arxiv.org/ abs/1603.01055.