Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Predicting the cost of second price auctions in a real time bidding environment

Tesis de Licenciatura en Ciencias de la Computación

Santiago Hernández

Director: Esteban Feuerstein

Buenos Aires, 2020

PREDICTING THE COST OF SECOND PRICE AUCTIONS IN A REAL TIME BIDDING ENVIRONMENT

Real-time Bidding (RTB) es una forma de gestionar espacios publicitarios que permite subastar espacios individuales mediante subastas en tiempo real, que suelen ser con la modalidad de sobre cerrado con segundo precio, de forma programática en el acto y por lo tanto el precio final que deberá pagar el oferente no es conocido al momento de ofertar.

RTB permite hacer publicidad a medida; es decir que los anuncios se pueden servir a los consumidores de forma directa basado en sus características demográficas, psicográficas y comportamiento.

Para optimizar campañas de marketing digital mostrando anuncios obtenidos mediante el mercado, el comprador debe poder valuar las oportunidades de mostrar anuncios y determinar si son rentables o no. Para esto es necesario una estimación del valor que aporta mostrar un anuncio y el costo de mostrar el mismo.

En este trabajo se introducen y comparan métodos para estimar el costo de subastas de segundo precio desde el punto de vista del comprador.

Estos métodos incluyen dos anteriormente públicados:

1. Una regresión lineal sobre el costo con un dataset de subastas ganadas.
2. Una regresión logística para estimar la probabilidad de ganar una subasta y luego aplicar métodos numéricos para estimar el costo aprovechando la relación entre ambos valores.

Y métodos aqui introducidos:

3. Una regresión logística sobre la proporción entre el costo y la oferta.
4. Una regresión logística para estimar la probabilidad de ganar una subasta y luego aplicar métodos analíticos para estimar el costo analogamente al método 2.

Los experimentos realizados muestran que el método introducido de utilizar una regresión logística sobre la proporción entre el costo y la oferta obtiene mejores resultados que el resto.

Palabras clave: estimación, costo, win rate, real time bidding, statistical learning, machine learning, subasta, subasta vickrey.

PREDICTING THE COST OF SECOND PRICE AUCTIONS IN A REAL TIME BIDDING ENVIRONMENT

Real-time Bidding (RTB) is a way of transacting media that allows an individual advertisement slot to be sold in real-time through a programmatic auction usually of the sealed bid second price kind and thus the final price the bidder has to pay is unknown at bidding time.

RTB allows for Addressable Advertising; the ability to serve ads to consumers directly based on their demographic, psychographic, or behavioral attributes.

In order to optimize a digital marketing campaign showing ads through rtb inventory, the buyer needs to be able to value advertisement opportunities and determine if they are profitable or not. To do this, an estimate of the value that showing the ad will provide and the cost of showing it are required.

In this work different approaches to predict the cost of second price auctions from the buyer's perspective are introduced and compared.

This approaches include two previously published methods:

1. A linear regression on a won auctions dataset.
2. A logistic regression to estimate the probability of winning an auction given a bid price, followed up by a numerical method to estimate the cost taking advantage of a relationship between both.

And new approaches introduced here:

3. A logistic regression to estimate the ratio of cost to bid price.
4. A logistic regression to estimate the probability of winning an auction given a bid price, followed up by an analytical method to estimate the cost in an analogous way to method 2.

The experiments show that the introduced method of using a logistic regression to estimate the ratio of cost to bid price outperforms the others.

Keywords: cost, win rate, real time bidding, statistical learning, machine learning, auction, second price, vickrey auction, prediction.

AGRADECIMIENTOS

Agradezco a mi director, Esteban.
Agradezco a mi amigo, compañero y mentor, Carlos.
Agradezco a mi familia.
Agradezco a mi amigos.
Agradezco a todos los profesores, docentes y no docentes de la facultad.
Agradezco a la suerte por haberme cruzado tan buena gente en este trayecto.
Agradezco por haberme acompañado, enseñado, compartido y por haber hecho esto posible.
Un saludo especial para Andy, Nico y Santi.
Gracias.

A mi persona favorita.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

In this work we propose and compare methods to estimate the costs of second price auctions in a Real Time Bidding scenario from the point of view of a bidder.

Real Time Bidding is a way of transacting media that allows an individual advertisement slot to be put up for bid in real time. This is done through a programmatic on the spot auction, which is similar to how financial markets operate. RTB allows for Addressable Advertising; the ability to serve ads (digital media display, video, mobile, social) to consumers directly based on their demographic, psychographic, or behavioural attributes, at the impression level.

The main appeal of the real time bidding market is that advertisers can show their ads to the right people at the right moment, and pay according to their value. In contrast with classic advertisement mediums like newspapers or TV commercials, the advertisers have data about the person that is about to see an ad.

Food delivery companies value highly people about to order food at noon and airline ticket sellers value higher wealthy people about to buy a last time minute first class ticket than people who, probably, are just curious about how much a ticket to an exotic destination costs.

This results in advertisers valuing users differently and app owners trying to determine the value of their users so they can sell their slots for as much as possible to optimize their revenue (so as to include an appropriate reserve price).

The cost of the impression in a mobile app can then be said to come out of:

- The application in which the ad slot is being offered.
- The context in which the slot is being offered. E.g.: the weekday, day's hour, geographic location.
- The owner of the phone about to see the ad.
- The interested buyers.

Most exchanges sell the slot to the highest bidder, charging the second highest bid or if there was a single bid, a pre-define floor (the minimum price and allowed bid).

More sofisticated exchanges can also offer first price auctions, or banning ads from competitors inside an app, for instance, a newspaper showing ads might choose to avoid showing ads of other newspapers.

1.1 Motivation

In order to optimize a digital marketing campaign using exclusively ads from real time bidding inventory, the buyer needs to be able to value advertisement opportunities and determine if they are profitable or not.

In order to evaluate or estimate the profitability of showing an ad to a certain person at a certain time, the buyer needs an estimate of the value that showing the ad will provide, and the cost of showing it.

As an example, let's assume that we have the opportunity to show an ad to a user, that after seeing the advertisement, will take a $100 ride to the airport, without considering the riders salary and the gas, if showing the ad costs less than a $100, we have a profitable opportunity, if it costs $100 or more, showing the ad is not profitable.

Another case would be having a chance to bid for two different ads to the same user, one for a train ticket and another one for a pizza. Assuming the train ticket ad has an expected net revenue of $5 and the pizza ad of $6, but also that this were the only moment to offer the train ticket, while there will be many opportunities in the near future to sell the pizza to this user, the DSP might choose to show the train ticket ad right now, and in the future show the pizza one.

Given that we have a valuation, we would like to estimate the cost of showing the ad, as it is sold through a second price auction, the final price might be lower than our offer, thus making the opportunity more profitable. This work's goal is providing estimations of the final price of second price auctions.

Determining the value of showing an ad to a user has been studied in other works, some of which will be introduced in the next section, and will not be further studied in this thesis.

In the case of Demand Side Platforms, which manage several digital marketing campaigns at the same time for different clients, they buy and pay ad slots to show its customers' ads, and later charges them using a certain criterion. Usual ones are charging per obtained click, install, in-app event or the advertisers return of investment [3]. This results in a disparity between how the DSP pays for showing ads and how it charges their customers based on results.

DSPs can have more than one customer interested in showing an ad to a certain user, in this case the DSP has to choose one of the customer's ads, this creates an internal competition or optimization problem inside the DSP. By considering the value, which will be different for each customer, the final price, and the opportunity cost for each of them, the DSP can choose which ad to bid for.

By smartly choosing when to bid, how much and for which of its customers, the DSP can optimize their revenue and their clients budgets.

1.2 Challenges

Online advertisement has risen to be a multi billion dollar industry and one of the main monetization strategies for free content providers on the internet. Given its importance and opportunities, companies have been reluctant to share their data, methods and how they use them to optimize their processes.

As a consequence of this, the publications related to this area are not as extensive as they could be. While there are multiple papers about Click Through Rates prediction, other types of conversion rates, winning rates and costs, have been less explored openly.

In this work we researched the prediction of second price auctions' costs. We used as a starting point publications and our own experience on the Click Through Rate prediction problem, a similar problem on the industry which has been more thoroughly studied than second price auctions' cost prediction.

In RTB digital marketing, online advertisement slots are sold without human interaction by different systems agreeing on buying and selling the slot at a certain price. As the decisions have to be taken automatically by a system expected to optimize the purchased

inventory given the amount of ads shown online, the amount of different variables to take into account and the fast response times needed to pick an advertisement, the field is an ideal environment to apply machine learning for fast and appropriate decision taking.

1.2.1 Detailed Description of the Problem

Time constraints

Once an ad slot is available, an application wants to keep the rendering of an ad fast, as long times impact the app's user experience and makes the users unhappy.

This results in time constraints in order to bid. Most exchanges require bidders to reply to auctions in under 120ms-300ms, including round trip time [15, 33, 32, 35].

Feature space cardinality

Each advertisement slot available is an opportunity with very diverse characteristics, when a bidder receives an auction offering slot, it is receiving a slot to show an ad to:

- A person that is N years old that identifies with the gender G,
- holding a phone from the year Y with the unique device identifier X,
- of the B brand, model M running the operative system O with a screen resolution R,
- using the social network application S,
- accesing the internet through a network type N of the carrier C, with the ip I
- at the H hours on the city C of the country K.
- And the slot is for a type T ad of size L with aspect ratio A.

Not all auctions will provide all the data, some might not provide the user's age, while others might provide other information, like the user's session time.

And this is only the data provided in each particular auction, besides this, the bidder might incorporate extra information stored using the device identifier as a key. The bidder could have extra information such as:

- Is this device available for showing ads recurrently?
- Does the user click on ads often?
- What other kind of applications do I know that the device has installed?
- Is this device used for making purchases regularly on other apps?

This leads to potentially having billions of variables available for consideration on a dataset with a few weeks of auctions and bids.

Amount of data

DSPs receive auctions from many sources and then evaluate each one of them for their customers. In order to fulfill the marketing requirements of their clients a lot of opportunities have to be evaluated to satisfy them all.

Some DSP have declared receiving between 500.000 and 1.000.000 auctions per second, like Criteo, Avazu or Jampp [12, 4, 5].

This means that after an hour, a DSP might have seen more than 1800 million auctions, assuming a 1% bid rate (only bidding in 1 percent of those auctions, which is not a farfetched number), this translates to 18 million bids per hour. If 10% of those auctions are won and the ad is shown, the bidder will have shown 1.8 million ads and will be notified how much has to be paid after the auction.

This means a DSP's RTB second price costs dataset can easily grow by more than 40 million observations per day. Also, if the DSP wants to incorporate data from all the bids, which contain information not only about winning cases, but also about bids in which the auction was lost, the DSP can have more than 430 million observations per day considering cases in which it knows its bid was too low to win, and therefore the cost was higher.

While having a vast amount of data can be considered good, it also brings its disadvantages, as working and processing huge datasets is more difficult than small ones.

Changing market

Demand, supply and prices can change very fast and often, the real time bidding environment is an always changing market.

Simple examples of this are important dates like black friday in the western world, single's day in China or christmas, during this days, advertisers increase their marketing budgets considerably to get more exposure, this results in more bids to satisfy them, and on higher bids as the opportunities become more valuable as users are more like to convert due to the offers and the general commercial spirit of those dates. This results in a more competitive market, where demand is higher and at higher prices, from one moment to other.

Another case is when an app becomes popular or loses popularity. When an app suddenly gets an important user base and shows ads the right way, demand for its slots can go very high, until the initial spark calms down, people might start using it less, get used to their ads and start ignoring them, and demand goes down.

This dynamism makes important to adapt to fluctuations and new supply in the market fast, as one might end up overpaying to show ads, optimizing campaigns suboptimally or losing the auctions and not getting enough inventory.

The real-time bidding (RTB) participants

The auctions are facilitated by ad exchanges as intermediate between publishers / *SSPs* (Supply Side Platforms) and bidders / *DSPs* (Demand Side Platforms).

In figure 1.1 we can see 5 key players:

- Advertisers: app owners that want to advertise their apps.
- Audience: regular people using mobile phones which will be shown advertisements.
- Publishers: mobile apps that show ads.

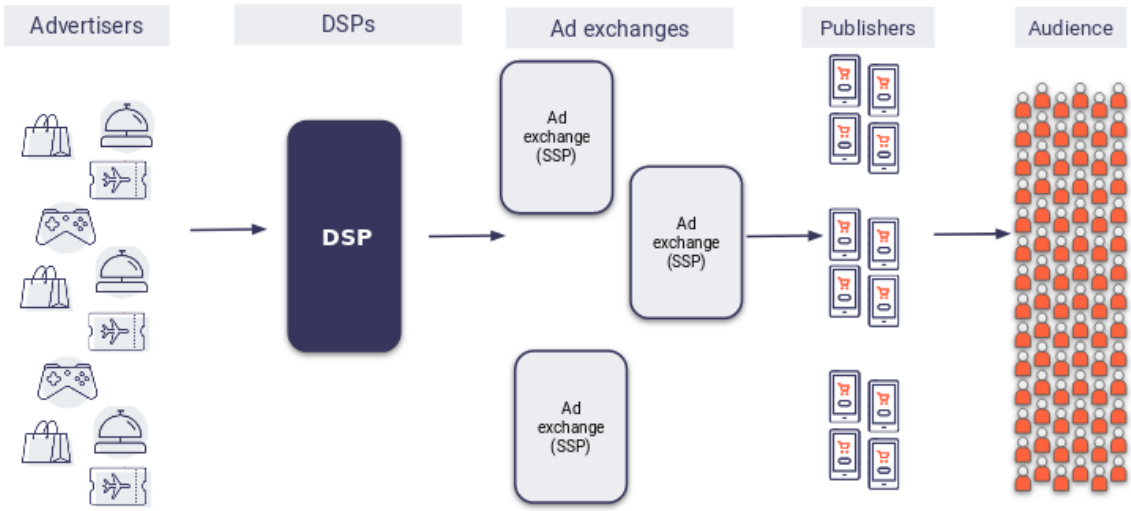

Fig. 1.1: Relevant players in the real time bidding ecosystem

- A DSP (Demand side platform): this are platforms whose customers are advertisers, that are in charge of programmatically advertising their app.
- Ad exchanges (SSP, Supply side platform): ad exchanges connect publishers with DSPs by receiving ad slot opportunities from the publishers and offering them in auctions to DSPs.

We are going to focus on a DSP. A DSP's architecture has been explained in [2]. Its main goal is to buy opportunities to show advertisements on behalf of its clients in order to get them new users or to encourage the current users to make a purchase in the client's application. An example of this would be showing an ad of a food ordering app at noon.

The real-time bidding (RTB) cycle

Here is a simplified overview of the Real time bidding cycle to sell media spots in mobile apps, as shown in figure 1.2:

1. A person is using a mobile app as usual in his phone.
2. There is an opportunity to shown an ad inside the app.
3. The app communicates this to an Ad exchange which operates as an auction house.
4. The exchange starts an auction and communicates it to several potential buyers (known as DPSs, Demand Side Platforms).
5. The DSP then decides to bid at a certain price to show an ad or not to bid at all.
6. Once all the bids are collected, the exchange determines who won.
7. Then, it provides the ad to the publishing app in order to show it to the user.

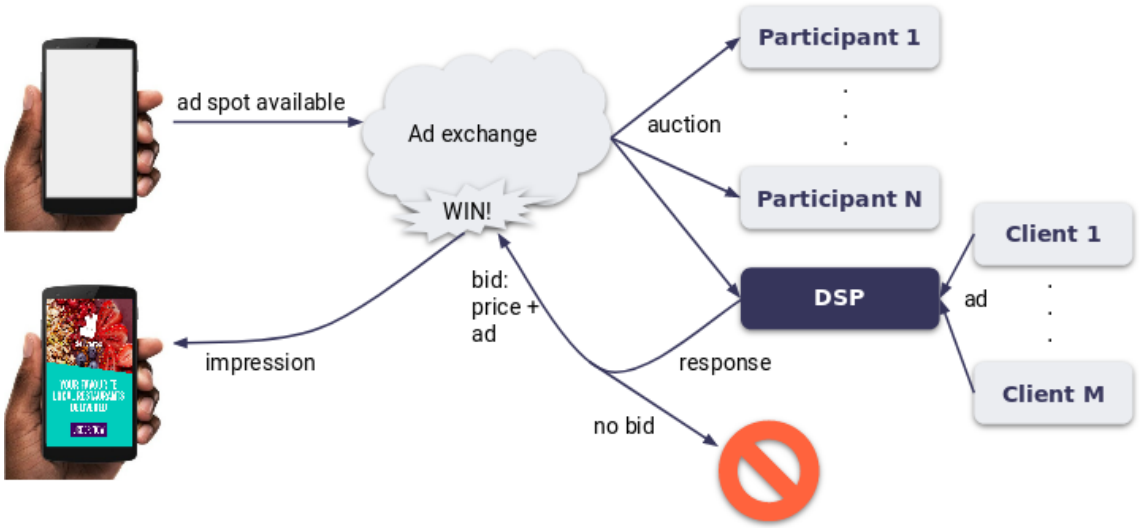

Fig. 1.2: Real time bidding cycle flow

8. And finally notifies the winner that the ad was shown (an impression), along with the price (as these are second price auctions, the winner does not pay its bid price, he pays what the second highest bidder bid or in case there was not another bidder, the auction's price floor).

Transaction lifecycle: possibilities after the impression

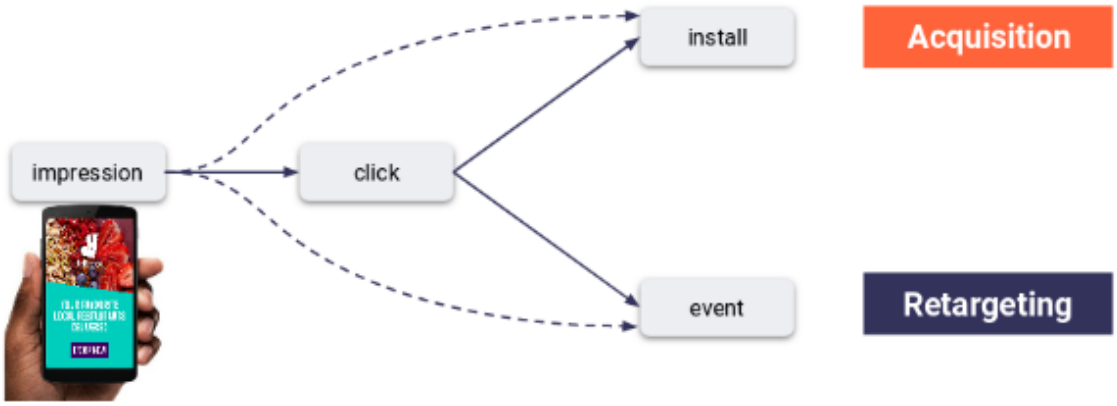

Fig. 1.3: Events posterior to the impression / click

Once an ad is shown, the user can click on it, what takes the him to either the App Store in order to download the app if he doesn't have it installed, or opens up the app otherwise.

If the app is installed as a result of clicking the ad, the DSP receives a notification indicating so.

In case the user performs events inside the app, like adding an item to a cart or purchasing something in an e-commerce, notifications are sent to the DSP.

This flow corresponds to the straight lines in figure 1.3.

Nowadays the industry is pivoting towards accepting view through attribution, which means that the ad shown will be recognized as the cause of a posterior event, even without having the user clicking on the ad. An example would be seeing a food delivery ad, not clicking on it, but still ordering food through the app 5 minutes later. This kind of attribution is represented by the dashed lines in figure 1.3.

1.3 Auction theory

Auctions have been used for a long long time as a mechanism to sell items and in very diverse markets, from Dutch merchants selling flowers, governments selling treasury bills, art owners selling master pieces, to more recently, online advertising opportunities. Just sponsored search in internet search engines has grown to a business with over 30 billion dollars a year in revenue.

In this section we give an overview of the basic types of auctions, strategies to participate in them and implications about these, including sealed first price, sealed second price, English and Dutch auctions.

The information in this section was compiled from [22, 25, 47].

1.3.1 Basic auction models

There are four main basic kinds of auctions used for selling single items. Even though the way they take place is very different, it can be seen that under some general conditions they all generate the same expected revenue, this is known as revenue equivalence.

Sealed bid auctions

In sealed bid auctions, an item is put up for sale, possibly having a minimum *reserve price*, and bids are sent simultaneously in a sealed way by the participants, meaning that they are only aware of their own bid and not of the other participants's ones.

The highest bidder wins the auction.

First price auction

In first price auctions, the winner pays his bid price for the item.

Second price auction

In second price auctions, the winner pays the second highest bid price for the item (plus maybe a small increment, like a cent), or in some cases, the *reserve price* if he was the only bidder.

Open auctions

In open auctions, after an item is put up for sale, all the participants are aware of the other participants bids.

English auction

English auctions can be considered the *classic* kind and the kind of auctions most people associate the word *auction* with. It is also portrayed in several movies and TV shows such as *North by Northwest (1959)*, *The Magic Christian (1969)*, *First Wives Club (1996)*, *Batman & Robin (1997)*, *Sex and the City (2008)* and *The Best Offer (2013)*.

This kind of auction begins by an auctioneer offering an item starting at a minimum *reserve price*, and then accepting increasingly higher bids from participants. Participants have a given time frame to bid a higher price than the current offer, until no participant bids higher than the current one. The winner of the auction is the participant who bid the highest price and pays his/her bid for the item.

While the auction takes place, all the participants know the current highest bid.

English auctions that last for a given time interval have been used at Ebay.com to sell items on their platform since the mid 1990s.

Dutch auction

In this kind of auctions an auctioneer begins asking a high price for an item, and sequentially lowers it in steps, until one of the participants accepts buying the item for the current price. The price might be lowered up to a minimum *reserve price*.

This auctions are famous for being used in the dutch flower market starting in the 17th century. This are also used by the United States Department of Treasury, through the Federal Reserve Bank of New York to raise funds for the United States government.

1.3.2 Bidding strategies

Second price auction

Since this work is about predicting the price of second price auctions, we will begin with this kind.

We will now show that in second price auctions, every bidder has a dominant strategy (a strategy that always maximizes the bidder's utility, no matter what other bidders do): set his bid b_i equal to his private valuation v_i. This results have been introduced by William Vickrey in [48] and in the following we explain why this is a dominant strategy.

Given an auction in which the bidder i has a valuation v_i for the item (this is fixed and cannot be changed), we have $B = max_{j \neq i} b_j$ (the highest bid of the other bidders), then the utility of the bidder i is the $valuation_i - paid\_price_i$ when he wins the auction. In case he loses, he doesn't have to pay the item, but also doesn't win the item of valuation v_i, getting an utility of 0.

We want to see that the bidder i's utility is maximized by bidding $b_i = v_i$.

In the second price auction, there are two possible outcomes for a bidder, either he has the highest bid, wins and pays B, or he doesn't and losses.

If $B > b_i$, the bidder i losses the auction. In this case the bidder does not get the item and the utility is 0.

If $b_i > B$, then the bidder i wins the auction, and its utility is $v_i - B$.

In this context, we can consider two cases:

1. If $v_i < B$, the highest utility the bidder can get is $max(0, v_i - B) = 0$. Which implies that in order to not have a negative utility, the bidder should get a utility of 0, by bidding at most v_i and losing in the first place, this is bidding truthfully.

2. If $v_i \geq B$, then its utility is $max(0, v_i - B) = v_i - B$. As $b_i > B$, B is paid, and that utility is achieved bidding truthfully.

As in both cases the utility is maximized by bidding truthfully, and by bidding truthfully in the second case it is also ensured that b_i is the value with the highest odds of winning the auction that yields a positive utility, we can say that bidding truthfully is a dominant strategy.

This implies that second price auctions are easy to take part in, as the bidders do not need to consider the other bidders or their bids, to determine what to bid.

Assuming every bidder in a second price auction bids truthfully, which is a reasonable assumption given that it guarantees non-negative and optimal utilities, then the item is given to the bidder who values it the most.

First price auction

What to bid in first price auctions is harder to determine than in second price auctions, supossing that the bidder i knew the bids of all the other bidders, what would the best bid be?

We can answer that question based on the analysis from the second price auction, we could say that if $v_i < max_{j \neq i} b_j$, then the bidder should lose the auction, in order to get an utility of 0. If $v_i > max_{j \neq i} b_j$, then $b_i = max_{j \neq i} b_j + \epsilon$, in order to maximize $v_i - b_i$. In colloquial language this means not taking part in the auction if our valuation is smaller than the highest bid and bidding the highest bid plus a small increment if our valuation is higher than the other participants' bids.

As a bidder does not actually know the other bidders' bids, then it has to somehow determine a bid price by considering the tradeoff between bidding closer to its valuation, potentially getting a smaller utility but with higher odds of winning, or bidding further away from its valuation, getting worse odds of winning the auction but with a potentially higher utility.

English auction

In an English auction, the bidder should keep participating in the auction until the price reaches his/her value, until that moment his/her utility is positive and after that, it turns negative.

As the highest bidder will not have to increase his bid after the second highest bidder reached his value and stopped bidding, the price of the item will be the second highest bid (plus an small increment), resulting in the price equivalent to the second price auction's.

In this kind of auctions, bidding up until one owns valuation is a dominant strategy too, as it is in second price sealed auctions. This auctions are sometimes referred to as open second price auctions.

The fact that the auction is open and that along the bidding process bidders gain information about the other bidders' valuation, could alter some of the bidders' valuations or behaviour, making english auctions differ more from second price sealed auctions.

Dutch auction

In a dutch auction, in order to obtain a positive utility, the bidder should not bid for an item as long as the price is above his valuation. Once the price drops belows its valuation,

the bidder can decide whether to bid for it at the current price or risk getting a cheaper price but with the possibility of another bidder bidding before him. This tradeoff between the worse odds of getting the item along with a larger utility is equivalent to the one in first price auctions.

Due to this relationship, dutch auctions are sometimes referred as open first price auctions, and a bidder could use the same strategy to bid in both of them.

1.3.3 Revenue equivalence

A specific case of the revenue equivalence theorem was first introduced by Vickrey in 1961 in [48] and the general case was introduced in 1981 by Myerson in [34] and Riley and Samuelson in [40]. The importance of this theorem relies on how it helps auction designers understand different auction mechanisms and create better ones. Auction design, the revenue equivalence theorem and recent applications of new auction mechanism have been put together by Paul Milgrom in [31].

The revenue equivalence theorem states that:

Any auction mechanism in which, for n risk-neutral bidders, each has a privately known value drawn independently from a common, strictly-increasing, atomless distribution.

Then, any such mechanism, in which:

1. the object always goes to the bidder with the highest bid
2. any bidder with the lowest-feasible bid expects zero utility

yields the same expected revenue (and results in each bidder making the same expected payment as a function of her signal).

This has important implications, mostly from the auctioneer's point of view:

1. The four basic auction kinds seen above yield the same expected revenue.
2. If the auctioneer wants to increase the expected revenue, the auction's mechanism has to be modified, for instance, by including a reservation price on the item (or minimum disclosed bid). In the analysis above we could think of this reservation price as another bid.
3. With risk-averse bidders, instead of risk-neutral as required by the revenue equivalence theorem, first-price auctions generate more revenue than second price auctions. Second-price auctions are not affected by risk-aversion as there is a dominant bidding strategy.

In this context, risk-averse bidders would opt for bidding prices closer to their valuation, getting a higher chance of winning the auction (with a potential smaller utility), on the other hand risk-seeking would opt for bidding lower prices in an attempt to increase the utility, even if the expected utility is the same as a risk-averse bid. A risk-neutral bidder is neither risk-averse nor risk-seeking, and is indifferent about the uncertainty of the options in a set of outcomes, and would not have a preference among two options with the same expected revenue. E.g.: a risk-neutral bidder would not have a preference between a bid that results in a 20% chance of winning a $10 utility, and a bid that results in a 50% chance of winning a $4 utility, as the expected utility of both are $2.

1.3.4 Second price auctions in Real Time Bidding

Second price auctions, also known as Vickrey auctions in honor to William Vickrey who first described and studied this type of auctions in a published work in 1961 [48], consist in a single auction for an item in which bids are provided "sealed", meaning that bids are known only to each bidder, and the winner pays the second highest bid (plus one extra cent or increment depending on the exchange) as it has been explained above.

This auctions have the advantage that the optimal bidding strategy is simple, robust and dominant, it does not depend on the rivals, and it consists of bidding thruthfully, this means bidding the maximum price the bidder is willing to pay.

How one values the item and decides the maximum price one would pay, is another story and could be complex.

Even though it was thought that Vickrey auctions had been rare, it was found out that this type of auction had been used since 1893 by stamp collectors and by other small groups since then [28]. Nowadays Vickrey auctions, or variations of them, are the preferred method of selling media spaces for advertising campaigns in websites and mobile applications, with billions of auctions per second.

A few extra variations about how media spaces are auctioned have been introduced, as that current exchanges have added floors as minimums values for the bids, or that second price auctions can turn into first price auctions for a specific bidder, if it has a private arrangement with the owner of the space being offered. In this work we will assume that the auctions have floor prices, as most of them do, and that we are not bidding with private deals that turn second price auctions into first price ones.

1.4 Basic bidding strategy

Given an auction for an ad slot, the DSP has a set of customers interested in it. For each client, the DSP has a goal the client wants to obtain (a click, an install or an event inside their application) by showing an ad to the user and an amount of money the client has arranged to pay the DSP for obtaining such goal. The DSP has to decide whether or not to bid, and in case it does, for which customer, for which of its ads and the bid price to offer.

As a final note, we are considering second price auctions in which each player only know his/her own value, therefore it can be proved using game theory that the optimal decision is to bid your valuation on the item.

As there is extensive literature and previous work on how to estimate this rates (e.g.: [30, 19, 39, 21, 17, 27, 60, 11, 10, 57]), we are assuming the DSP already has estimators for:

- The Win Rate: the probability of winning an auction by bidding an amount of money b.
- The Conversion Rate: the probability that the user will click, install the application or perform a certain event in it, given that the ad is shown, or that it is shown and clicked.

Here we will portray a bidding strategy algorithm to select the customer, the ad and

the bid price to offer in an auction.

Algorithm 1: Bidding strategy to optimize the expected net revenue

Input: auction
Input: customers: set of customers for the auction
Result: The customer, ad and bid combination that optimizes the expected net revenue

```
1  best ← None
2  foreach customer ∈ customers do
3      foreach ad ∈ customer.ads do
4          expected_conversion ← predict_conversion(auction, customer, ad)
5          expected_revenue ←
             expected_conversion * customer.payment_per_conversion
6          bid_price ← CalculateBidPrice(customer, expected_revenue)
7          expected_win_rate ← predict_win_rate(auction, bid_price)
8          expected_cost ← predict_cost(auction, bid_price)
9          expected_net_revenue ←
             expected_win_rate * (expected_revenue − expected_cost)
10         if best is None or expected_net_revenue > best.expected_net_revenue then
11             best ← (customer, ad, bid_price)
12 return best
```

We will go line by line explaining algorithm 1 to optimize the expected net revenue of an auction for a DSP.

1 Initialize the best option as null.

2 Iterate over all the DSP's customers with marketing campaigns matching the auction (e.g.: select the customers running campaigns on Brazil when an auction from Brazil is received).

3 Iterate over the customer's ads. As it might have more than one ad (e.g.: the same ad with different background colors or showing different products).

4 Estimate the probability of the user converting after seeing the ad (e.g.: estimate the probability of the user taking a taxi after seeing a taxi ad). The referenced papers on CTR prediction cover the case for click prediction.

5 We define the expected revenue as what the DSP gets paid for getting the customer's event of interest (which could be a click, an app install, an in app purchase, or other in app event), considering the estimated probability of such event happening. This is the expected value for the DSP of showing the ad.

6 We calculate the bid price from the expected revenue (the value) and the customer. In this step the DSP returns the action's expected revenue (its valuation) as one would expect for a second price auction. In some cases, the DSP might consider adjusting the bid considering the pacing (how fast it is buying ads for the client) and its opportunity cost. There could also be hard and soft business requirements (e.g.: in extreme cases one might want to offer more than its valuation, risking a monetary loss, in order to keep a client). This topic won't be studied in this work.

7 Once a bid price is defined, one can estimate the win rate, i.e. the probability of winning the auction.

8 Once a bid price is defined, one can also estimate the cost of the second price auction (the goal of this work).

9 The expected net revenue, the profit the DSP expects to make from this auction, can be calculated this way, the expected revenue minus the cost, considering the probability that the ad is shown and the subsequent events can take place.

10-11 Keep the customer, ad and bid price that has the maximum expected net revenue.

12 Returns the ad and the bid price that optimizes the expected net revenue.

As it can be seen, having an estimation for the cost allows us to choose the customer that maximizes the expected net revenue.

We should note that there are many potential strategies a DSP can use to bid and optimize their business, however it is not the goal of this work to make a survey of them, and most likely this strategies will benefit of having a cost estimation.

1.5 Statistical learning

As we want to estimate the cost of second price auctions given input variables, we will use statistical learning which comprises methods to learn from data and make numerical predictions.

Statistical learning [18] is an area in statistics which provides tools for collecting and analizing datasets. Faster computers, larger memory and more powerful algorithms now allow to work with datasets larger than what classic statistics ever dealt with. Most of the algorithms studied in statistical learning are also studied in machine learning, therefore many of the methods we talk about in the thesis can be found in both areas.

This whole section is strongly based on and takes much from [20, 18].

Given the characteristics of the problem described in the *Challenges* section, it seems naive to define hard set rules to code a program that predicts the cost of an auction based on our own domain knowledge.

For this reasons is that we would like to have an algorithm or piece of software able to analyze data from previous auctions, "learn" the rules and then estimate new auctions' costs based on them. This kind of algorithms are called regression algorithms in the *supervised learning* area of study in statistical learning and machine learning.

By using a regression algorithm, we can get a system able to process data of billions of auctions on real time, determine how auctions' costs vary depending on the available variables in our dataset, and then use this output to estimate the second price cost of new auctions.

1.5.1 Supervised learning

In supervised learning, the problem at hand consists of learning from a dataset in which we have input values, also called predictors, independent variables or features, and its associated output value(s), also called dependant variable, response or in some cases, label. And then use the learned model to predict the output value of certain input values.

Variables can be quantitative (also called numerical), like a person's height, or qualitative (also called categorical), which take the value of one of k different categories, like a person's country of origin, or "positive or negative".

We have a training dataset formed by a set of observations, each of which have one or several input values, and the set of output values corresponding to the observations.
X: input values
Y: output values

Where X_i is a vector containing the features of the i^{th} observation in our dataset, and Y_i is a vector (or single value) with the output value of the i^{th} observation. Assuming that there is a relationship between X and Y, such that we can write $Y = g(x) + \epsilon$ where ϵ is a random error term, which is independent of X and has zero mean. We want to estimate g using X and Y, such that we can use the estimated Y, $\hat{Y} = \hat{g}(X)$, where $\hat{g}$ is an estimation of g, in a predictive setting.

Regression and classification

When the problem consists of estimating a numerical variable or variables, it's called a regression problem.

On the other hand, when the problem consists of estimating the value of a categorical variable or variables, it is called a classification problem.

In this work we will focus on a regression problem, as we want to estimate the winning price of a second price auction, a numerical variable, from a set of input variables. Nevertheless, we will tackle this problem by modelling it with both regression and classification approaches.

1.5.2 Parametric methods

In order to estimate g we will use parametric methods, this involves a two step model based approach.

1. First, we make an assumption about the functional form of g, for instance, we could assume that g is linear in X, i.e.:

   $g(x) = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$

   In this case g has a linear model.

2. After a model has been selected, we follow a procedure to fit the model from the training data, this is finding the value of the model's parameters. Notice that depending on the selected model, the procedure can be more complex or simpler.

Linear regression

Linear regression is a classic approach for predicting a quantitative response, in which it is assumed that there is approximately a linear relationship between X and Y. This relationship where Y is approximately modeled can be written as:

$$Y \approx \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$$

The β_i are known as the model's coefficients or parameters, and after using the training data to estimate them, we can predict a new y from an x input, by doing:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \ldots + \hat{\beta}_n x_n$$

Where the caret symbol means they are estimations.

Logistic regression

Logistic regression is one of the most widely used approaches to solve classification problems with a binary outcome, normally labeled as positive or negative, using 1 and 0 values. The usual problem tackled with logistic regression is that we have a positive class and a negative class, input variables and we want an estimation of $P(output = positive|input\ variables)$. Often, once there is an estimated probability of being positive, the output is classified as positive if the estimated probability is above a certain threshold.

More generally, we want to model the relationship $P(Y = 1|X)$, whose values should be between 0 and 1.

In logistic regression, we use the logistic function to model $P(x)$:

$$p(X) = \frac{e^{\beta_0+\beta_1\ x_1+\cdots+\beta_n\ x_n}}{1+e^{\beta_0+\beta_1\ x_1+\cdots+\beta_n\ x_n}} = \frac{1}{1+e^{-(\beta_0+\beta_1\ x_1+\cdots+\beta_n\ x_n)}}$$

After some manipulation, it can be seen that:

$$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1\ x_1+\cdots+\beta_n\ x_n}$$

Where $p(X)/(1-p(x))$ is called the odds and can take any value between 0 and ∞. By taking the logarithm on both sides we get the log odds or logit:

$$log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1\ x_1 + \cdots + \beta_n\ x_n$$

It is noticeable that the right hand is linear in X.

Once the coefficients have been estimated, we can get a class probability estimation by doing:

$$\hat{p}(X) = \frac{1}{1+e^{-(\hat{\beta_0}+\hat{\beta_1}\ x_1+\cdots+\hat{\beta_n}\ x_n)}}$$

1.5.3 Estimating the coefficients

Least squares for linear regression

The most common approach to estimating a linear regression's coefficients involves minimizing the least squares criterion.

Let $\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_n x_{in}$ the prediction for Y on the i^{th} value of X.

Then the residual is defined as $e_i = y_i - \hat{y}_i$, this is the difference between the i^{th} response and the i^{th} estimated output value.

The residual sum of squares (RSS) is $RSS = e_1^2 + \cdots + e_n^2$.

The least squares approach chooses the β_i parameters that minimize the RSS.

Maximum likelihood for logistic regression

The most common approach to estimating a logistic regression's coefficients involves maximizing the likelihood function:

$$L(\Theta) = \prod_{i \in \{1,\cdots,N\}, y_i=1} P(y = 1|x = x; \Theta) \cdot \prod_{i \in \{1,\cdots,N\}, y_i=0} P(y = 0|x = x; \Theta)$$

Where Θ denotes the parameters of the model. Notice that for the i^{th} observation this is: $y_i * log(\hat{p}(x_i)) + (1 - y_i) * log(1 - \hat{p}(x_i))$.

Alternatively, instead of maximizing the likelihood function, the negative log likelihood is often minimized: $-log(L(\Theta))$.

1.5.4 Model comparison and selection

Once we have models to solve our problem, we want to be able to compare them and select one, particularly in a predictive problem, we want a selection method to choose the model that provides the best estimations against new data without requiring using them all in a controlled experiment.

There are several approaches to model evaluation and selection, like hold out validation [38], cross-validation and bootstrap [23, 59], structural risk minimization [44], bayesian information criteria [42] or the Akaike information criteria [1].

In the problem faced in this work, the data is intrinsically ordered, as win rates and auction's prices vary highly across time depending on the market's supply and demand, this makes classic cross-validation approaches to have its pitfalls for model evaluation, and therefore we will appeal to hold out and sliding window cross validation [6, 46], training on a set of contiguous days and testing on the next one, and for following folds, sliding the days' window.

In order to perform validation, we have to select and compare a metric or error.

As it is explained in [43], "There are (at least) three ways we can use statistical models in data analysis: as summaries of the data, as predictors, and as simulators.

With any predictive model, we can gauge how well it works by looking at its errors. We want these to be small; if they can't be small all the time we'd like them to be small on average. We may also want them to be patternless or unsystematic.

Because our models are flawed, we have limited data and the world is stochastic, we cannot expect even the best model to have zero error. Instead, we would like to minimize the expected error, or risk, or generalization error, on new data.

What we would like to do is to minimize the risk or expected loss

$$\mathbb{E}[L(Z, \Theta)] = \int L(z, \Theta)p(z)dz$$

For mean squared error, this would be:

$$L(z, \Theta) = (y - m_\Theta(x))^2$$

For mean absolute error:

$$L(z, \Theta) = |y - m_\Theta(x)|$$

To do this, however, we'd have to be able to calculate that expectation. Since we don't know the true joint distribution of X and Y, we need to approximate it somehow.

We will take the data and split it into training and testing sets.

Fitting to one part of the data, and evaluating on the other, gives us an unbiased estimate of generalization error." [43]

For our particular problem, as we are going to center our attention on evaluating the models using the hold out and sliding window validation methods, we will select three metrics, which will be described in the experiments section.

1.6 Gradient descent

Gradient descent is an iterative algorithm for finding the minimum of a function by taking steps proportional to the negative of the gradient. If the function is convex, it has a single global minimum and gradient descent can converge to it [52].

The basic gradient descent procedure is as follows [13]:

Algorithm 2: Basic gradient descent algorithm

Input: v: parameters vector
Input: f: criterion function to minimize
Input: α: learning rate
Input: tol: tolerance stopping criteria
Result: Parameters vector that minimize the function f

1 **repeat**
2 $\quad v \leftarrow v - \alpha * \nabla f(v)$
3 **until** $\alpha * \nabla f(v) < tol$;

The basic gradient descent approach has many known problems such as setting the learning rate, adjusting the learning rate to improve the optimization (a small learning rate can mean a needlessly slow convergence, while a too big one can even make it diverge), or defining a stopping criteria. These have been tackled in different ways and many of this methods are covered in [41].

Loss functions for linear and logistic regression

As we presented in the statistical learning section 1.5, linear and logistic regression's parameters are selected to minimize the residual sum of squares and maximize the likelihood function respectively. We also presented the negative log likelihood for logistic regression, an alternative criterion that is to be minimized, not maximized, and therefore can be directly used with the gradient descent method to estimate the parameters of the model. Therefore we want to minimize the residual sum of squares or the negative log likelihood loss functions to estimate the coefficients of the linear and logistic regressions using gradient descent.

Ideally we would like to minimize the *expected loss*, this is minimizing the loss given any input-output pair. As the input-output probability distribution is not available at our disposal, the loss cannot be measured against it. However we can measure the empirical loss against our input-output samples dataset, obtaining the *empirical loss*.

1.6.1 Batch vs online gradient descent

The usual batch method of finding the parameters of a linear or logistic regression performs one parameters update after the average $\nabla f(v)$ is calculated for the whole training dataset, as it is shown in algorithm 3. In the method known as online gradient descent (or stochastic gradient descent), $\nabla f(v)$ is calculated for every single observation and the parameters updated after that. In some cases the observation being processed is randomly selected from the dataset and in others it is selected sequentially, along with consuming the whole

dataset as the stopping criteria. The basic algorithm is shown in Algorithm 4.

Algorithm 3: Basic batch gradient descent algorithm for linear/logistic regression

Input: v: parameters vector
Input: d: dataset sample
Input: f: criterion function to minimize
Input: α: learning rate
Input: tol: tolerance stopping criteria
Result: Parameters vector that minimize the function f

$n \leftarrow size(d)$
repeat
  $v \leftarrow v - \frac{\alpha}{n} * \sum_{i=1}^{n} \nabla f(v, d_i)$
until $\frac{\alpha}{n} * \sum_{i=1}^{n} \nabla f(v, d_i) < tol$;

Algorithm 4: Basic online gradient descent algorithm for linear/logistic regression

Input: v: parameters vector
Input: d: dataset sample
Input: f: criterion function to minimize
Input: α: learning rate
Input: n: amount of fitted samples stopping criteria
Result: Parameters vector that minimize the function f

$k \leftarrow 0$
repeat
  $k \leftarrow k + 1$
  $o \leftarrow uniform\_random\_sample\_one(d)$
  $v \leftarrow v - \alpha * \nabla f(v, o)$
until $k \geq n$;

1.6.2 Convergence

As it is very well explained in [8] and reproduced below, the amount of computations, information used from the dataset and parameters updates performed per iteration for batch and online methods have big impacts on their convergence:

Batch convergence

A batch approach can minimize the empirical loss R_n at a fast rate. If R_n is strongly convex for a batch gradient method there exists a constant $\rho \in (0, 1)$ such that $\forall k \; in \; \mathbb{N}$, the training error satisfies:

$$R_n(w_k) - R_n^* \leq O(p^k)$$

Where R_n^* is the minimal value of R_n and w_k are the parameters at update k. Then the total work required to obtain ϵ-optimality is proportional to $n \; log(1/\epsilon)$. n because each update requires to calculate n gradients, one for each observation. $log(1/\epsilon)$ because in the worst case its the total number of iterations in which the training error can be above a given $\epsilon > 0$.

Online convergence

A basic online approach has a slower convergence rate than batch approaches. If R_n is strictly convex and each i_k observation is drawn uniformly from $\{1, \ldots, n\}$ for each parameters update, then, for all $k \in \mathbb{N}$:

$$\mathbb{E}[R_n(w) - R_n^*] = O(1/k)$$

It can be seen that for the online approach neither the per-iteration cost nor $O(1/k)$ depend on the sample set size n. The total work required to achieve ϵ-optimality is proportional to $1/\epsilon$.

Comparisson

While $n\ log(1/\epsilon)$ can be smaller for moderate values of n and ϵ, when one is working with large datasets and is limited by computational time, the online approach's $1/k$ is favored.

This will be taken into consideration when selecting a method to fit our estimators.

1.7 Related work

We can build our solution facing the problem's challenges resting on previous work that together can answer the following question:

How can we process a big amount of data points, which are generated on real time, containing potentially billions of features that are not known in advance, whose distribution might fluctuate from one moment to the other?

And then use this information to make estimations fast?

In [7], an analysis on processing large scale datasets, shows advantages on using algorithms with slower convergence that are able to process large amounts of data on a limited amount of time, over using algorithms with faster convergence but that are slower. While on [29] several algorithms are presented and analyzed.

In [50], a method known as the hashing trick to handle the dimensionality of the feature space is explained.

In [30], a practical case of such algorithms to predict click rates in digital advertising is shown.

This publications are the basis used for this work's implementation and we will look at them in further detail.

Large-scale machine learning

In [7], stochastic gradient descent for optimization algorithms is analyzed on small vs large datasets. The main premise of this work is that on the last decades, the datasets sizes have grown faster than the speed of processors. Due to this, we have reached an inflection point in which statistical learning problems can be devided in two:

- Small scale: the ones in which the amount of observations in the dataset is the limiting factor, as the number of examples is not enough to make the computing time an issue. In this cases the optimization and estimation errors can be minimized by defining the threshold at which we want the optimizer to stop and by choosing the amount of observations from the data to use. In this case, the approximation-estimation tradeoff studied in statistical learning arises.

- Large scale: the ones in which computing time is the limiting factor. In this cases, approximate optimization can achieve better expected loss because more training examples can be processed on the same amount of time. One could choose an algorithm with faster asymptotic convergence processing less data or for longer time, or an algorithm with slower asymptotic convergence processing more data in the same amount of time.

Bottou proves that algorithms with worse asymptotic convergence, like stochastic gradient descent or second order stochastic gradient descent take less time to reach a predefined risk than algorithms with faster asymptotic convergence, like gradient descent or second order stochastic gradient descent. This is due to the computing time of each iteration of the stochastic versions being considerably faster than the non-stochastic ones, allowing them to process more data in the same amount of time.

On the practical level, the paper reports running several optimizers on a dataset, in which the stochastic algorithms reach the best test performance in a couple of minutes, while the standard CRF L-BFGS optimizer took 72 minutes.

This study is very relevant to our problem as we are working on a large scale learning task and therefore we will look into using online optimization algorithms to be able to process larger datasets in restricted amounts of time.

Follow the regularized leader - proximal (FTRL-p)

In [29], the follow the regularized leader - proximal algorithm for online convex optimization applied to online learning is introduced. It is also shown an equivalence interpretation between mirror descent algorithms for online convex optimization, such as online gradient descent, and follow the regularized leader. The difference between this and other algorithms is how it handles regularization terms. In this publication it is shown with a real world dataset how FTRL-p outperforms the other studied algorithms.

The main advantage of this approach is that large dataset can be processed for classification or linear regression problems efficiently and producing sparse models.

Considering how we have to process large data and that the dataset can contain billions of features, an algorithm such as FTRL-p to handle the data size and to produce sparse models out of billions of features sounds ideal, and is the one we will be using. The algorithm itself will be explained 1.8.

Click through rates prediction

In the digital advertising industry, click through rates prediction is a classic problem and there are several publications about it.

This arises mostly from two reasons:

1. Clicks on ads have been used (and still are on some cases) as a way to measure the success of advertisement, as they are relatively easy to track and are considered an indicator of interest from the user on the product.

2. Charging a fixed amount of money per click to clients running marketing campaigns is a popular spend model in the digital advertising industry, due to the reasons from the previous point, and because it splits the risk among the client and the provider. The provider has to ensure that it is showing the ads to people interested in the product, but it does not take the risk of spending money to show ads that might not result in a sale. On the other side, even though a positive return of investment is not guaranteed as a sale might not take place, the client has less risk compared to the model where it pays for every ad shown, regardless of the interest of the user in it.

As a result of this, DSPs have given a lot of importance to being good at predicting click through rates. This problem in this industry is characterized for having millions of data points with dozens of categorical variables and many numerical ones with relevant variable interactions, potentially adding up to billions of features.

In [30] the experience of deploying a production system for CTR prediction at Google Inc. using the FTRL-p algorithm is presented claiming great results.

In [19] a CTR prediction system at Facebook is presented, in which the best results are obtained by:

- Generating categorical features from numerical features through gradient boosting trees.

- Using a logisting regression with stochastic gradient descent with per-coordinate learning rate (such as FTRL-p has).
- And keeping the estimator up to date (data freshness matters).

In [37], a presentation by Criteo, it was shown how they chose an approach using logistic regression over Field-aware Factorization Machines, due to the computation performance of both algorithms.

These two papers further motivates the use of a logistic / linear regression model optimized with the FTRL-proximal algorithm, while also introducing the importance of keeping estimators up to date.

Feature hashing for large scale multitask learning

In [50] the hashing trick is studied. This consists of hashing a high dimensional input vector into a lower dimensional feature space. This method preserves sparsity, introduces no additional overhead of storing projection matrices or dictionaries, and pre-determines a feature space in which any new feature can be projected to, through a hash function.

In the paper it is shown theoretically and empirically how the hashing trick can be used to obtain impressive classification results on real-world problems, and that random subspaces of the hashed space are likely to not interact.

This method is ideal for a large scale learning task with billions of features without a constrained space, as it both reduces the dimensionality of the problem and pre-defines a feature space, allowing to define the size of a parametric method's model such as linear or logistic regression before hand. The hashing trick method will be explained in 1.8.

Second price auction cost prediction

The publications on estimating the costs of second price auctions are not many as in the CTR prediction case.

In [26] it is presented how Drawbridge, an ad-tech company, estimates win rates and winning prices of second price auctions on RTB. In this work they compare linear regressions on won auctions costs and doing numerical integration on win rates prediction based on logistic regression, due to a dual relationship between both problems, to predict the auction's cost. Both approaches will be compared in this work, along with two new approaches, and will be further explained in the models section.

In [56] a linear regression model is used on won auctions data and it is ensembled with a censored regression model using both won and lost auctions data. The linear regression model is used on won auctions only, as it performed better in that case.

In [58], survival rate analysis approaches are used on both won and lost auctions to estimate the market price distribution, taking advantage of the win rate and impression cost relationship as in [26].

1.8 Algorithms

In this section we will introduce two algorithms that we will use to address the machine learning task at hand, that attack two major challenges in this problem:

1. Training an estimator with datasets large enough to not fit in RAM memory, while supporting high feature dimensionality, achieving sparse models in relation to the feature space, all while getting a good performance.

2. The feature generation and feature space dimensionality.

In [7], it was noted that for large scale machine learning problems, in which the computation time is the limitation and not the amount of data, there are tradeoffs between the algorithms used and the amount of data processed in a given time window. Specifically, it has to be taken into account that given a limited amount of time, and a simpler family of functions to fit with faster algorithms, the overall error can be reduced more than with complex functions, as the simpler one would be able to process more data, reducing the optimization error.

In order to tackle this two problems, taking into consideration that we are facing a large scale machine learning problem, we will use the follow the regularized leader proximal algorithm to fit a logistic regression along with the hashing trick.

1.8.1 Follow The Regularized Leader - proximal

From the necessity of fitting an estimator in a memory efficient way, fast, using large scale datasets and generating sparse models, is that we draw on online convex optimizations algorithms to fit our estimators.

In [29], the most successful algorithms that meet those requirements were studied and the FTRL-proximal algorithm was proposed and compared against the others in a CTR dataset. The FTRL-proximal logistic regression got the best performance while being the most sparse, through the use of lambda 1 regularization.

After that publication, in [30], the challenges and approaches to tackle CTR prediction, a massive-scale learning problem of similar characteristics to the one studied in this work, were thoroughly discussed. It was also noted the success of the FTRL-proximal logistic regression to obtain great results while keeping the final model sparse.

As it has been explained in both [30] and [29], the FTRL-proximal algorithm is the same as online gradient descent when no regularization is included. However, once regularization is added, FTRL-proximal moves from the online gradient descent weight update, given a sequence of gradients g_t:

$$w_{t+1} = w_t - \eta_t g_t$$

Where η_t is a non decreasing learning rate.
FTRL-proximal updates the weights as:

$$w_{t+1} = \text{argmin}_w(g_{1:t} \cdot w + \frac{1}{2}\sum_{s=1}^{t} \sigma_s ||w - w_s||_2^2 + \lambda_1 ||w||_1)$$

Where η_s is defined in terms of the learning rate, such that

$$\sigma_{1:t} = \frac{1}{\eta_t}$$

Even though it may seem that this optimization algorithm is harder to implement and that it is required to keep the gradients' history, it is not required to do so, and it can be

implemented keeping one extra number per feature. An example implementation based on [30] is shown in this section's listing.

```
from math import exp, sqrt


def sign(x):
    return 1 if x >= 0 else -1


class Linear:

    def ilink(x):
        return x

    def dloss(pred, label):
        return pred - label


class Logistic:

    def ilink(x):
        return 1 / (1 + exp(max(-100, min(-x, 100))))

    def dloss(pred, label):
        return pred - label


class Estimator:

    def __init__(self, alpha=0.04, beta=0.1, lambda1=0, lambda2=0,
                 max_feat_number=2**22, regression=Logistic):
        self.alpha = alpha
        self.beta = beta
        self.lambda1 = lambda1
        self.lambda2 = lambda2
        self.z = [0.0] * max_feat_number  # sum of grads - sigma-weighted
    ↪ coefs
        self.n = [0] * max_feat_number  # number of times feature seen
        self.w = [0.0] * max_feat_number  # reusable weights vector
        self.regression = regression

    def fit(self, indexes, values, label):
        """Fits a single observation.

        Parameters
        ----------
        indexes: List[int]
            A sparse list containing the indexes of the non zero elements in
    ↪  a
            numeric vector representing an observation.
        values: List[float]
            A sparse list containing the numeric values of the corresponding
            index in the same position of indexes.
        label: numeric
            The numeric ground truth value
        """
        n, z, w = self.n, self.z, self.w
        pred = self.estimate(indexes, values)
```

```
55          dloss = self.regression.dloss(pred, label)
56          for i, value in zip(indexes, values):
57              g = dloss * value  # gradient loss w.r.t. wi
58              sigma = 1 / self.alpha * (sqrt(n[i] + g ** 2) - sqrt(n[i]))
59              z[i] = z[i] + g - sigma * w[i]
60              n[i] = n[i] + g ** 2
61          return pred
62
63      def estimate(self, indexes, values):
64          return self.regression.ilink(self.project(indexes, values))
65
66      def project(self, indexes, values):
67          a, b = self.alpha, self.beta
68          l1, l2 = self.lambda1, self.lambda2
69          z, n, w = self.z, self.n, self.w
70          wTx = 0
71          for i, value in zip(indexes, values):
72              if l1 <= z[i] <= l1:
73                  w[i] = 0
74              else:
75                  w[i] = (- ((b + sqrt(n[i])) / a + l2) ** -1 * (z[i] -
76                            sign(z[i]) * l1))
77              wTx += w[i] * value
78          return wTx
```

Listing 1.1: Example implementation of the logistic regression FTRL-proximal

1.8.2 Hashing trick

As we will show in the *Datasets' Variables* section and on the Models' *Base model* section, this problem from the advertising technology industry is characterized for potentially having billions of features, while each observation is sparse, usually having non zero variables in the order of the hundreds.

For this reason is that it would be particularly useful to have a way of turning the features of our model into a numeric vector representation that is memory efficient, easy to compute, can handle arbitrary new features and does not require an in-memory mapping of variables to numeric representation.

One way to solve this problem, and the one we will use in our experiments, is the method known as the hashing-trick which has shown great success in practice [30, 50].

This consists of taking a hash of a numeric or string variable and using the output value as the index in a numeric vector representing the observation.

The vector's size is often smaller than the potential amount of features, fitting any arbitrary feature space into a limited feature space.

This implies that:

1. If we want to keep track of the semantic meaning of each position, we will have to keep an inverse hash table from hashes / indexes to features.

2. There might be collisions when assigning an index in the vector, and if we want to keep an inverse hash table, we will either have to keep all the values that generate the same hash, and / or define an strategy to pick one.

In Vowpal Wabbit [24], an open source fast machine learning system, the hashing trick is implemented by taking a feature name, hashing it using a hash function and then taking the modulo $1 << b$ to limit the amount of bits of the index to the desired range. For quadratic features combining two of them, the first feature's hashed index is multiplied by a constant, then xored against the second feature, and the modulo is taken again. This procedure allows to generate deterministic vector indexes from any amount of categorical or numeric variables, and generate a sparse vector representation, without the need of storing an in memory mapping.

In "Feature Hashing for Large Scale Multitask Learning" [50], an implementation is suggested containing two hash functions, one to determine the index of the feature into the vector, and another binary one, mapping the variable into $\{-1, 1\}$ to remove bias and fight hash collisions, by multiplying the value in the feature vector by ± 1 in the corresponding index coming from the first hash function. Then, in order to have a collision, the values of both hash functions have to collide.

A third implementation we used, is hashing the variables and keeping a second vector with the amount of times a feature has appeared. Then, on the estimator, we keep track of the most seen feature in a given index using a majority vote online algorithm, like Boyer-Moore's algorithm [9], to determine the most relevant feature in an index, in case there are collisions.

```
from functools import reduce
from operator import mul
from typing import List, Tuple, Union

from xxhash import xxh64


def hash_function(x):
    return xxh64(x).intdigest()


def hash_feature(feature_names: Tuple[str],
                 feature_values: Tuple[Union[str, float]]):
    """Returns a feature hash and its value.

    Parameters
    ----------
    feature_names: Tuple(str)
        A tuple of strings containing the feature names.
        E.g.: ('country', 'day_of_week', 'bid')
    feature_values: Tuple(Union(str, float))
        A tuple with the values of the corresponding feature names.
        E.g.: ('argentina', 'monday', 1.4)

    Returns
    -------
    Tuple(int, float)
        The hash of the feature and its numeric value
    """
    # Sort the tuples acording to the feature name to ensure that the hash's
    # output is independent from the tuple's input order
    feature_names, feature_values = zip(*sorted(
```

```
        zip(feature_names, feature_values),
        key=lambda name_value: name_value[0]))
    # Turn numeric values into a string representation to ensure that the
    # feature's hash output is independent from the variable's numeric value
    stringified_values = [value if isinstance(value, str) else 'float'
                          for value in feature_values]
    # For categorical variables, set the value as a 1
    numeric_values = [1 if isinstance(value, str) else value
                      for value in feature_values]
    # Join all the sorted feature names and their values to hash them
    hashed = hash_function(
        '+'.join(feature_names) + ',' + '-'.join(stringified_values))
    # The value of a polinomial feature as the multiplication of the values
    value = reduce(mul, numeric_values)
    return hashed, value


def hashing_vectorizer(features_names: List[Tuple[str]],
                       features_values: List[Tuple[Union[str, float]]],
                       n: int):
    """Returns a feature hash and its value.

    Parameters
    ----------
    features_names: List(Tuple(str))
        A list of tuples of strings containing the feature names.
        E.g.: [('country', 'day_of_week', 'bid'), ('hour_of_day')]
    features_values: Tuple(Union(str, float))
        A list of tuples with the values of the corresponding to features
    ↪ names
        E.g.: [('argentina', 'monday', 1.4), (3)]
    n: int
        The maximum index value allowed in the vector

    Returns
    -------
    Tuple(List(int), List(float))
        The hash of the feature and its numeric value
    """
    indexes = []
    values = []
    for feature_names, feature_values in zip(features_names, features_values
    ↪ ):
        index, value = hash_feature(feature_names, feature_values)
        indexes.append(index % n)
        values.append(value)
    return indexes, values


features_names = [('country', 'day_of_week', 'bid'),
                  ('bid_floor',),
                  ('operative_system', 'country')]
features_values = [('argentina', 'monday', 1.3),
                   (0.4,),
                   ('android', 'argentina')]
n = 1 << 22
indexes, values = hashing_vectorizer(features_names, features_values, n)
print(indexes)
```

```
89 print(values)
90
91 [1226380, 3947233, 3831511]
92 [1.3, 0.4, 1]
```

Listing 1.2: An example implementation of the hashing trick

1.9 Costs and win rate prediction

As we can see and as it has been discussed in [26] and [58], the win rate and cost prediction problems are related. By definition we have that:

$$win\ rate = P(cost \leq bid)$$

Then by the definition of cumulative distribution function and probability density function, we know that given a random variable X, be F_x its c.d.f. and f_x its p.d.f., then:

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(t)dt$$

Replacing the values:

$$F(bid) = P(cost \leq bid) = win\ rate = \int_{-\infty}^{bid} f(t)dt$$

Now we can see that the win rate is the c.d.f of the p.d.f of the auction's cost.

This implies that if we have a win rate distribution, we can take the derivative w.r.t. to the bid, we have the cost distribution, and if we have a cost distribution, we can integrate it up to the bid, to get the win rate. Knowing this, if we can solve the integral or the derivative, we could solve both problems with only one estimator.

In the models section we will explain how we can exploit this relationship to get cost predictions from a win rate estimator, and on the datasets section, why would we want to do this.

Win rate prediction

The win rate estimation problem consists of a classification problem in which we have to determine if an auction will be won or not. This is analogous to the click through rate problem.

New approaches proposed

In this work we extend the win rate predictions approach doing analytical integration using a closed form formula instead of numerical integration as in [26], and introducing approaches to estimate the ratio of the cost to the bid on a new dataset of won auctions only.

The auction's cost prediction can be used in the bidder's strategy to optimize their bidding decisions.

1.10 Our contribution

In this work we compared different approaches for predicting the cost of second price auctions in a real time bidding environment:

1. A linear regression on the cost of previously won auctions.
2. Numerically integrating a logistic regression on the probability of winning an auction.
3. A logistic regression on the ratio of the cost to the bid price.
4. A logistic regression on the ratio of the cost to the bid price starting from the auction's bid floor.
5. Analytically integrating a logistic regression on the probability of winning an auction.
6. Three simple approaches that do not require any previous data as baselines:
   (a) Predicting the auction's bid floor.
   (b) Predicting the bidder's bid.
   (c) Predicting the aritmethic mean between the floor and the bid.

While the first two approaches had already been studied in the literature, the other ones are introduced here.

Stochastic online gradient descent optimized linear and logistic regression algorithms along with the hashing trick were used to fit a dataset with million of data points corresponding to two weeks worth of time changing data from a DSP.

The results obtained show that the forth approach outperforms the others, that the analytical integration outperforms the numerical integration, and that the third simple approach can be more convenient and obtain better results than the first two.

This could allow for better cost predictions on second price auctions, which can be used to improve the bidding strategies of DSPs, which could help improve their net revenue and their clients' return of investment. At the same time smaller datasets than the ones required for the second method could be used.

1.11 Thesis' structure

Having introduced the basic concepts and motivation for this work, we will we will proceed to describe how data is gathered, its complications and possible solutions.

Next, we will continue presenting the different estimators we are going to evaluate.

Finally, the experiments will be presented, along with the results, discussion and conclusion.

2. DATA

In order to build our datasets, we use the information from the *auctions*, *bid* and *ad shown* messages, which share a common transaction id which allows us to join them together.

Auction messages provide information about the ad slot that is being offered. In bid messages, the offered bid price is included. Finally in impression (ad shown) messages, the cost the bidder has to pay is informed.

	Auction data				Bid data	Impression data	
Id	Time	Country	Ad type	...	Bid	Ad shown	Cost
1	2018-07-14 16:51	Argentina	Html	...	0.23	0	NULL
3	2018-07-14 16:51	Japan	Video	...	0.33	1	0.24
9	2018-07-14 16:52	Spain	Image	...	0.12	1	0.03
...	...	...	...	...	...	...	...

Tab. 2.1: Example of a dataset showing the origin of the variables

As bidders are only notified if they win the auction and the ad was shown, in order to label negative cases, we have to wait a reasonable amount of time until we are sure that a delayed notification won't arrive, which implies several difficulties as described in [36]. This has two potential problems:

1. If we don't wait enough time before the case is labeled as negative, we can label cases as negatives and afterwards receive the win notifications, ending up with false negatives in our datasets.

2. The longer we wait for win notifications before labelling cases as negative, the more out of date our estimators will be. E.g.: if we wait 20 minutes, the most recent data in our dataset will be from auctions from 20 minutes ago. If we wait 2 hours, the most recent data will be from 2 hours ago. Therefore, in order to be the most up to date as possible, we want to wait as little as possible for labelling, while not having many false negatives.

In table 2.1 we can see an example of how the dataset is built by joining information from the different stages of the real time bidding auction flow. Notice how in the first row, in which the ad was not shown, there is no cost and in the other cases, the bid is greater or equal than the final price.

2.1 Variables

Some of the the variables that might be available for extraction are:

- Ad slot size (in pixels): for image or video ads, the ad slot's width and height is provided.

- Ad slot tag id: Identifier for the ad placement inside the app / web.

- Ad slot type: image banner, html banner, native (disguise your ad as a native element of the app / web, e.g.: like another post), video or audio.
- Application in which the ad slot is available.
- Application category: e.g. social network, tourism, arts and entertainment, etc.
- Application paid: boolean flag indicating whether the app is free or a paid version.
- Bid floor: the minimum acceptable bid and the amount of money the bidder will be charged in a single bidder auction, in dollars.
- Bid price: the amount of money bidded in dollars.
- Browser's user agent: name of the user agent used to render the web / app.
- Carrier: name of the carrier providing service to the user. E.g.: AT&T, Movistar, Claro, T-Mobile, etc.
- Coppa: Flag indicating if this request is subject to the COPPA (Child Online Privacy Protection Act) regulations established by the USA FTC.
- Country, region and city, latitude / longitude: the user's current latitude and longitude, from which the country, region and city can be derived, or directly this values.
- Day of the week: the current day of the week.
- Device type: the kind of device the user is using the app in. E.g.: mobile/tablet, personal computer, tv, phone, tablet.
- Do not track: flag indicating if the user does not want to be tracked.
- Exchange: the name of the exchange running the auction.
- Is interstitial: boolean flag indicating if the ad covers the whole page before showing the content of the app / web.
- Mobile device's model: e.g. Samsung Galaxy S3.
- Mobile device's operative system: e.g.: Android 4.3.1.
- Network type: if the user is accessing the internet through a wi-fi, cellular gsm, g3, g4, etc. connection.
- Time of the day: timestamp providing the time of the day.
- Traffic type: whether the ad slot is in an application or web.
- User's age and gender.
- Video / audio min / max duration: the minimum and maximum duration allowed for video / audio ads.
- Video skippable: boolean flag to indicate if the video ad is skippable.

While a few variables are numeric, like the bid and the bid floor, most are categorical and it has to be taken into account that in order to use categorical variables in a linear or logistic regression, some kind of *hashing* or *one-hot encoding* has to be performed on the variables. This kind of encoding will transform a single categorical variable with multiple values into one binary variable per value.

Also regarding categorical variables in this particular problem, many do not even have a limited amount or predefined set of possible values. Taking for instance the application in which the slot is offered, new applications are uploaded to the App Store everyday, while many are taken down.

It is also important to note that there are relevant variable interactions (e.g.: the country with the time of the day).

This things result in an amount of possible features on the order of the billions, if we take for instance as a feature the day of the week on a given country, we have 7 possible days and 193 possible countries, spanning to 1351 possible combinations, if we now combine this with the application in which the ad would be shown, we would have 1351 combinations with almost 3 millions apps in the Google PlayStore, of which more than 95% are free [16] (although not all of them have in app ads), we can see that just combining these three variables could lead to over a billion features.

2.2 Datasets

2.2.1 Costs dataset

This dataset consists of one observation per ad shown and the labels are numeric values indicating the cost.

In table 2.2 we have examples from a costs dataset, in this case we want to train an estimator for the last column (second price cost), using the other columns as predictors. What we have are data from won auctions and we want to estimate the second price auction cost, given the bid price and the other variables.

Note: we show a few of the available variables for illustration purposes, but there could be more not portrayed here.

2.2.2 Win rate dataset

This dataset consists of one observation per bid and the labels are binary responses indicating whether the auction was won and the ad shown or not.

In table 2.3 we have examples from a win rate dataset, in this case we want to train an estimator for the last column (Won), using the other columns as predictors. What we have are data from auctions in which the DSP bid, and we want to estimate the probability of winning said auction, given the bid price and the other variables. Notice that all the rows for won auctions (i.e. Won is 1), will be available in the cost dataset with the second price cost, while lost auctions (i.e. Won is 0), won't be available in the other dataset, as the DSP is not notified how much the winner paid for the impression.

Note: we show a few of the available variables for illustration purposes, but there could be more not portrayed here.

Device maker	Model	Lang.	Bid floor	Carrier	Country id	Latitude	Longitude	Network	Platform	Region	Application	Exchange id	Ad size	Video ad	Native ad	Bid	Second price cost
Apple	iPhone	-	0.133	WIFI	178	14.471	121.022	wifi	IOS	-	Wattpad - iOS	121	300x250	False	False	0.46	0.33
Apple	iPhone	en	0.02	Sprint Nextel	237	39.037	-77.05	cellular	IOS	Maryland	MyFitnessPal	168	320x50	False	False	0.15	0.07
Apple	iPhone	en	0.02	AT&T	237	37.735	-122.373	cellular	IOS	California	TMZ - iOS Match Buy	168	320x50	False	False	0.12	0.05
Apple	iPhone	en	0.02	Verizon Wireless	237	40.442	-79.983	cellular	IOS	Pennsylvania	TMZ - iOS Match Buy	168	300x250	False	False	0.15	0.02
-	MotoG3	pt	0.15	WIFI	32	-23.551	-46.633	wifi	ANDROID	Sao Paulo	Palavra Guru	259	320x480	True	False	0.18	0.15
Apple	iPhone	en	0.011	AT&T	237	41.921	-87.704	cellular	IOS	Illinois	FOX59 News	168	320x50	False	False	0.11	0.04
motorola	Moto G (4)	pt	0.03	(WI-FI) TIM	32	-23.63	-46.632	wifi	ANDROID	Sao Paulo	Picross galaxy	272	320x50	False	False	0.03	0.03
Apple	iPhone	en	0.011	Other	237	36.028	-96.073	cellular	IOS	Oklahoma	The Weather Channel	168	320x50	False	False	0.14	0.12
Samsung	SM-G610M	pt	0.121	10	32	-23.546	-46.629	wifi	ANDROID	Sao Paulo	Color Road	44	320x50	False	False	0.17	0.13
AT&T	SM-J337A	en	0.02	Other	237	33.167	-87.506	cellular	ANDROID	Alabama	Trivia Crack	168	320x50	False	False	0.11	0.02
Samsung	SM-N920K	-	0.154	WIFI	120	37.51	127.107	wifi	ANDROID	Seoul-t'ukpyolsi	AlarmMon	121	300x250	False	False	0.4	0.29
Alcatel	5010G	en	0.02	Other	49	6.186	-74.997	cellular	ANDROID	Antioquia	Trivia Crack	168	320x50	False	False	0.14	0.06

Tab. 2.2: Example data from the second price cost dataset

Device maker	Model	Lang.	Bid floor	Carrier	Country id	Latitude	Longitude	Network	Platform	Region	Application	Exchange id	Ad size	Video ad	Native ad	Bid	Won
LG	LM-X210(G)	en	0.013	Other	237	39.19	-77.235	cellular	ANDROID	Maryland	ibis Paint X	168	320x50	False	False	0.12	0
Apple	iPhone	en	0.033	Other	237	43.073	-89.453	cellular	IOS	Wisconsin	MyFitnessPal	168	320x50	False	False	0.14	1
Apple	iPhone	en	0.02	T-Mobile	237	25.732	-80.26	cellular	IOS	North Carolina	MyFitnessPal	168	320x50	False	False	0.11	0
Apple	iPhone	en	0.02	Other	237	26.115	-80.367	wifi	IOS	Florida	TMZ - iOS Match Buy	168	300x250	False	False	0.1	0
-	SM-J105B	pt	0.15	WIFI	32	-21.132	-44.253	wifi	ANDROID	Para	Subway Surfers	259	320x480	True	False	1.14	1
Motorola	Moto G (5) Plus	-	0.143	WIFI	32	-23.63	-46.632	wifi	ANDROID	Sao Paulo	Whats Web	121	768x1024	False	False	0.19	0
Apple	iPhone	en	0.056	Other	237	41.121	-73.483	cellular	IOS	Connecticut	MyFitnessPal	168	320x50	False	False	0.15	0
Apple	iPhone	en	0.011	Other	237	40.696	-73.326	cellular	IOS	New York	New York Post	168	320x50	False	False	0.14	1
-	SM-G3589W	en	0.154	CenturyLink	237	44.94	-93.219	wifi	ANDROID	Minnesota	Nox Launcher	178	-	False	True	0.16	1
Apple	iPhone	en	0.011	T-Mobile	237	25.755	-80.229	cellular	IOS	Florida	NBA	168	320x50	False	False	0.13	0
Lenovo	A2016b30	pt	0.02	Other	32	-23.63	-46.632	wifi	ANDROID	Sao Paulo	My Talking Hank	168	320x480	False	False	0.07	1
-	MotoG3	pt	0.15	WIFI	32	-15.8	-47.864	wifi	ANDROID	Mato Grosso do Sul	Score! Match	259	320x480	True	False	1.18	0
Samsung	SM-G610M	-	0.161	WIFI	32	-22.786	-43.433	wifi	ANDROID	Rio de Janeiro	Mini Golf: Retro	121	320x50	False	False	1.62	1
-	RCT6873W42..	en	0.25	WIFI	237	25.762	-80.192	wifi	ANDROID	Florida	Subway Surfers	259	320x480	False	False	0.44	0

Tab. 2.3: Example data from the win rate dataset

2.3 Sizes

An important aspect of the problem is the cardinality of the datasets, on any given week, the DSP could receive over 700 thousand auctions per second, which results in ∼600 million bids and ∼72 million ads shown per day. This roughly translate to a bid rate lower than 1%, and a win rate close to a 12%.

The impact this has is that the costs dataset grows by 72 million observations per day, while the win rate one does so by 600 million observations per day.

2.4 Complications

Having described the datasets, we list complications we have found:

1. The win rate dataset's size will be greater than or equal to the costs' one.
2. The win rate dataset has information on lost auctions due to bidding a low price, while these observations will be lost on the costs datasets.
3. There isn't a constraint feature space, over 1 billion different features can be easily seen after a few weeks.
4. Demand, supply and prices can change very fast and often, the real time bidding environment is an always changing market, which results in up to date estimators being needed in order to have accurate estimations.
5. Lack of transparency from exchanges: auctions are not audited by bidders, therefore there is no guarantee that there are actually other bids that act as second highest bid, or that exchanges are not sending multiple auctions for the same ad slot, using increasing bid floors.

2.5 Workarounds

1. This is not necessarily a complication and has strong and weak points for both datasets. Having more data in the win rate, makes it more expensive to store and process if we want to take advantage of all of it, on the other hand, having fewer data on the costs dataset, could not be a bad thing, for the opposite reasons, and because it still has millions of new observations per day.
2. In order to reduce the bias towards auctions in which we use to win with lower bidding prices, we can explore the other cases by boosting our bid prices and paying more for those.
   This way we can have access to data points from all the types of auctions for both datasets, even though the win rate dataset will most likely still count with more relevant information.

Points 3 and 4 have been addressed in the *algorithms* section.

Point 5 feasibility cannot be solved or at least confirm that it happens in a consistent way without auditing the *exchanges*, without this possibility, we will assume that there could be noise in the dataset due to questionable practices from auction houses. Even though this possibility is being considered, one would expect partners not to commit fraud or take advantage using dishonest practices.

3. MODELS

3.1 Base model variables

We will define a base model consisting in a set of predictors independent from the bid and a set of predictors containing the bid. This predictor variables will be used in all the linear and logistic regression approaches. Even though we were not able to find publications advocating for the exact variables used here, from talking with people in the industry we take that this are not very far away from what many companies use.

The first ones will consist of an intercept and contextual variables related to the auction as described in the datasets section.

The second one will consist of the first ones interacting with the bid, e.g: (the bid, the bid in country 1, etc.).

The predictors independent from the bid are included to capture information as a more specific intercept inherent to the market.

Considering the interaction among variables, the abstract base model consists of:

1. Intercept
2. Variables
3. Interaction(variables, variables)
4. Interaction(variables, bid price)
5. Interaction(variables, variables, bid price)

Illustrating this, if we had the following variables available: ad slot size, app and country, we would have:

1. Intercept
2. Variables: ad slot size, app, country
3. Interaction(variables, variables): (ad slot size, app), (ad slot size, country), (app, country)
4. Interaction(variables, bid price): (ad slot size, bid price), (app, bid price), (country, bid price)
5. Interaction(variables, variables, bid price): (ad slot size, app, bid price), (ad slot size, country, bid price), (app, country, bid price)

Then from an observation from Argentina in the Fruit Ninja app for a 320x50 pixels ad slot bidding $1.5, the resulting features would be:

1. Intercept
2. Ad slot size: 320x50

3. App: Fruit Ninja
4. Country: Argentina
5. Ad slot size: 320x50 in App: Fruit Ninja
6. Ad slot size: 320x50 in Country: Argentina
7. App: Fruit Ninja in Country: Argentina
8. Ad slot size: 320x50 bidding $1.5
9. App: Fruit Ninja bidding $1.5
10. Country: Argentina bidding $1.5
11. Ad slot size: 320x50 in App: Fruit Ninja bidding $1.5
12. Ad slot size: 320x50 in Country: Argentina bidding $1.5
13. App: Fruit Ninja in Country: Argentina bidding $1.5

As a consequence of generating this features, our estimator would be able to detect, for example, the relevance of Fruit Ninja in Argentina vs other countries.
The final amount of features in each observation would be:

1. Intercept: 1
2. Variables: n
3. Interaction(Variables, variables): $binomial\ coefficient(n, 2)$
4. Interaction(Variables, bid): n
5. Interaction(Variables, variables, bid): $binomial\ coefficient(n, 2)$

Then we have:

$$amount\ of\ features = 1 + 2 * n + 2 * \frac{n!}{k!(n-k)!}$$

Where n is the amount of variables used, as described in the dataset section.
For instance, selecting 10 variables, we would extract 111 features per observation.

Finally the variables we are going to use on our model are:

- Ad slot size
- Ad slot position tag
- Application
- Bid floor
- Bid price
- Country

- Day of week
- Hour of day
- Network connection type
- Platform (IOS / Android)
- Region in the country
- RTB exchange

3.2 Domain based information variants

As we are working on second price auctions, given that we win the auction, the cost will always be between the auction's bid floor and our bid.

As the regression models might not capture this fact, we will also evaluate clipping their outputs to the range $[bid\ floor, bid]$.

3.3 Linear model on costs

This is one of the methods that has been studied previously.

This is defined as a linear regression on the base model trained with the costs dataset, in which the label is the actual cost.

In order to obtain a cost estimation, it is enough to use the estimator's predict on a new observation.

3.4 Logistic model on cost over bid

3.4.1 Approach 1: cost ratio

This is defined as a logistic regression on the base model trained with the costs dataset, but in this case, the label value will be the **cost/bid ratio**. This means that if the bid was 1.0 and the cost 0.60, the label would be 0.6, thus making the labels take values in $[0.0, 1.0]$.

Afterwards, in order to estimate the expected cost, the rate estimation returned by the estimator has to be multiplied by the bid.

Procedure:

1. Label the observations as $label = cost\ ratio = cost/bid$.
2. Fit a logistic regression model on the dataset.
3. Estimate the *cost ratio* for the new observation.
4. $cost\ estimation = estimated\ cost\ ratio * bid$.

3.4.2 Approach 2: cost ratio from the floor

The second approach is analogous to the first one, but estimating the ratio between (**cost** − **bid floor**)/(**bid** − **bid floor**). Once we can estimate this, we can estimate the cost by doing:

$$ratio\ estimate * (bid - bid\ floor) + bid\ floor$$

The advantage of this model, is that it captures by design the fact the cost will always be between the bid floor and the bid.

3.5 Win rate prediction plus numeric integration

This is one of the methods that has been studied previously.

This is defined as a logistic regression on the base model trained with the win rate dataset. The estimation would tell us how likely we are to win an auction by bidding a certain amount of money.

Afterwards, in order to estimate the expected cost, several win rate estimations will be obtained for different bid prices, and the expected cost will be obtained by doing the numeric integration on them using the chained trapezoidal rule.

Procedure:

1. Fit a win rate estimator on the win rate dataset.
2. When a new case is received, make several win rate estimations ranging from the lowest possible value, to our bid.
3. Use the trapezoidal rule to integrate the values from the previous point.
4. The cost estimation is the result from point 3

3.5.1 The trapezoidal rule

As explained in [55], the trapezoidal rule approximates the integral of a function by approximating the region under the function as a trapezoid and integrating that.

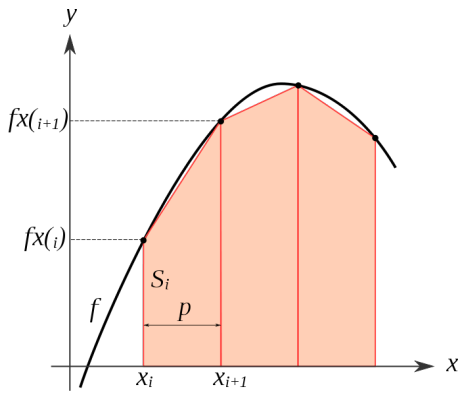

Fig. 3.1: Example of the chained trapezoidal rule.

The chained trapezoidal rule works by partitioning the integration interval and applying the rule in each part. In figure 3.1 we can see an example of this.

Therefore:

$$\int_a^b f(x)\,dx \approx \sum_{k=1}^{n} \frac{f(x_{k-1}) + f(x_k)}{2} \Delta x_k$$

```
def trapz(x, y):
    ''' Trapezoidal integration rule. Assumes y >= 0. '''
    squares = triangles = 0
    for i in range(1, len(x)):
        j = i - 1
        d_x = x[i] - x[j]
        squares += d_x * y[i]
        triangles += d_x * (y[i] - y[j])
    return squares - 0.5 * triangles
```

Listing 3.1: An example implementation of the chained trapezoidal rule

3.5.2 Amount of estimations and performance impact

As we want the final estimator to be as cpu performant as possible, we want to find the best tradeoff between predictive power and amount of estimations, by finding the smallest amount that we can use, without degrading the estimations too much.

3.5.3 Selecting bid values for the different estimators

As it can be seen in figure 3.2, most bids are concentrated on lower values, which translates on more training data for lower bids. In figure 3.3, we can see that the win rate follows a mostly linear increasing trend with respect to the bid price.

Based on this, we will consider two strategies to determine the bid values we are going to estimate on for the integral.

The first strategy will be selecting log spaced bid values between the minimum allowed bid and our own, to concentrate the values on the lower end.

The second strategy will be selecting uniformly spaced bid values between the minimum allowed bid and our own.

This is how we can implement the linear and log spaced values generation:

```
def linspace(start, end, points):
    ''' Equivalent to numpy's linspace '''
    delta = (end - start) / (points - 1)
    return [start + i * delta for i in range(points)]

def logspace(start, end, points, base=10):
    ''' Equivalent to numpy's logspace '''
    delta = (end - start) / (points - 1)
    return [base ** (start + i * delta) for i in range(points)]
```

Listing 3.2: Linear and log spaced value generation

Another way to look at this would be to take the derivative of the win rate distribution, which we know is:

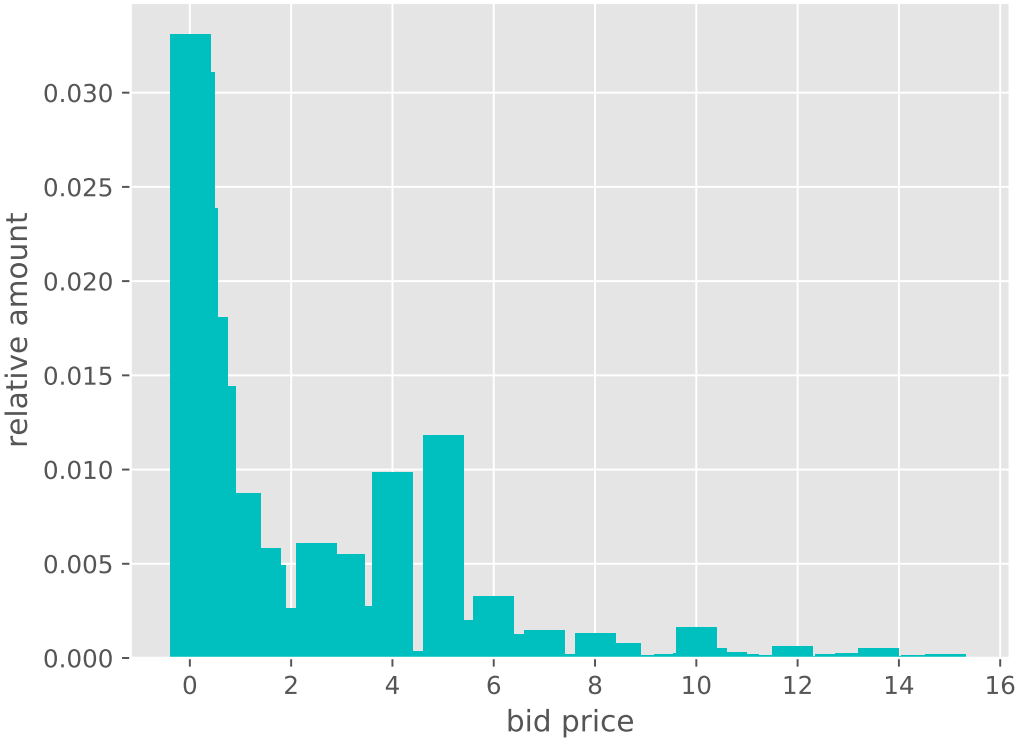

Fig. 3.2: Bid price histogram over a week worth of bids. Bids over $15 have been removed from the plot and constitute less than a 1% of the total amount of bids.

$$\left(1 + e^{-(\beta_0 + \beta_1 * bid)^{-1}}\right)$$

Then:

$$\frac{\partial\ (1+e^{-(\beta_0+\beta_1*bid)})^{-1}}{\partial\ bid} = \frac{\beta_1 * e^{-(\beta_0+\beta_1*bid)}}{(1+e^{-(\beta_0+\beta_1*bid)})^2} = f(x)$$

Which is the p.d.f of the cost distribution. Finally we get the expected cost by doing:

$$\mathbb{E}(x * f(x)), where\ x \leq bid$$

And there we could integrate $x * f(x)$ from 0 to the *bid*.

3.6 Win rate prediction plus analytic integration

The trained model and estimator will be the same as the one doing numeric integration, but the cost estimation method will change by calculating the integral analytically, which results in a closed form formula.

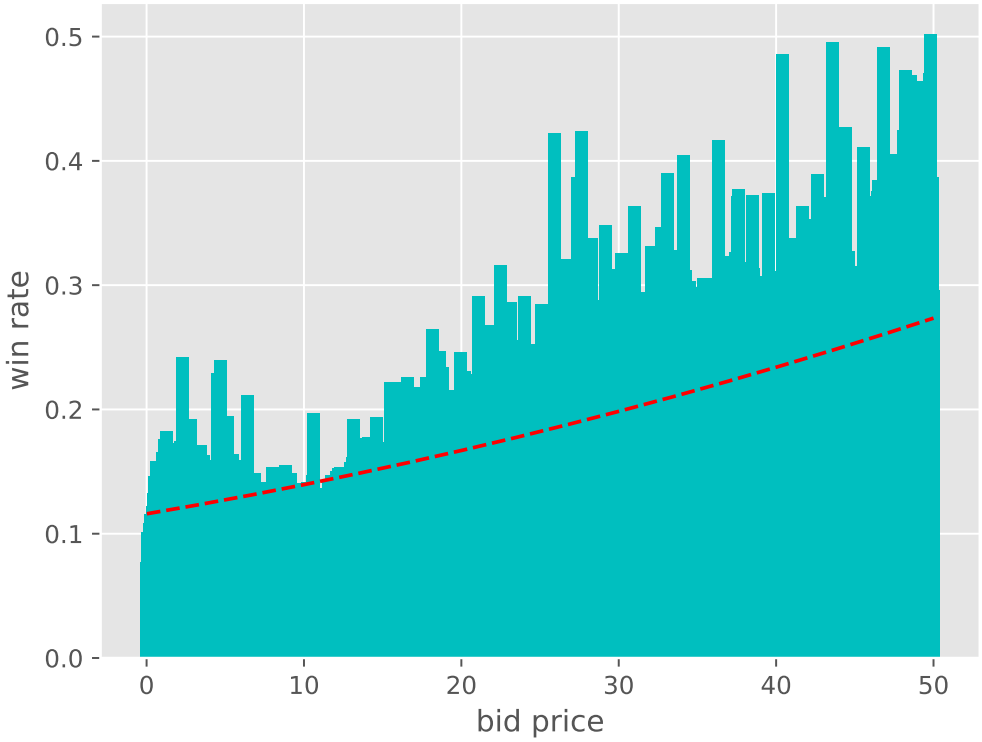

Fig. 3.3: Bid's win rate over a week worth of bids. The red dashed line shows the win rate trend.

The sigmoid function used in logistic regression is defined as:

$$sigmoid(x) = (1 + e^{-x})^{-1}$$

In this case x is the result of a dot product between two vectors, the regressors and the coefficients one. And we can divide the regressors vector in two:

1. Regressors independent from the bid
2. Regressors interacting with the bid

$$x = \beta_0 + \beta_1 c$$

Where:

$$c = bid$$

Therefore:

$$F_X(x) = (1 + e^{-(\beta_0 + \beta_1 c)})^{-1}$$

3.6.1 Integral part 1

Given X a continuous random variable and A an event with $P(A) > 0$. We have the following definitions:

C.D.F.:

$$F_X(x) = P(X \leq x)$$

P.D.F.:

$$f_X(x) = F'_X(x)$$

Expectation:

$$E(X) = \int_{-\infty}^{\infty} x \ f_X(x) \ dx$$

Conditional P.D.F. to an event:

$$f_{X|A}(x) = \begin{cases} \frac{f_X(x)}{P(x)} \ if \ x \in A \\ 0 \ otherwise \end{cases}$$

Expectation conditional to an event:

$$E(X|A) = \int_{-\infty}^{\infty} x \ f_{X|A}(x) \ dx$$

Lemma. If X is a continuous random variable and X is non-negative, then:

$$E(X) = \int_0^{\infty} P(X > y) \ dy = \int_0^{\infty} 1 - F_X(y) dy$$

3.6.2 Integral part 2

From the definitions above and knowing that $win\ rate = P(cost \leq bid)$ from the Cost and win rate prediction section.

We have that, x being the second price auction's cost and B the bid price:

$$E[x|x < B] = \int_{-\infty}^{\infty} x \ f_{X|X<B}(x) \ dx$$

Since X is non-negative, $f_{X|X<B}(x) = 0$ for $x \leq 0$:

$$E[X|X < B] = \int_0^{\infty} P(X > x|X < B) \ dx$$

$$= \int_0^{\infty} 1 - F_{X|X<B}(x) \ dx$$

$$= \int_0^{\infty} 1 - \frac{F_X(X < x \wedge X < B)}{F_X(B)} \ dx$$

As $0 < x \leq B$:

$$= \int_0^B 1 - \frac{F_X(x)}{F_X(B)}\ dx$$

$$= \int_0^B 1dx - \int_0^B 1\frac{F_X(x)}{F_X(B)}\ dx$$

$$= x|_0^B - \frac{1}{F_X(B)} \int_0^B F_X(x)\ dx$$

$$= B - \frac{1}{F_X(B)} \int_0^B F_X(x)\ dx$$

Substituting the variables names:

$$E[cost|cost \leq bid] = bid - \frac{1}{F_X(bid)} \int_0^{bid} F_X(c)\ dc$$

Where $F_X(x)$ is the probability of winning the auction given the bid price x.

3.6.3 Integral part 3

Now integrating $F_X(c)dc$.

$$\int_0^b F(c)\mathrm{d}c = \int_0^b (1 + e^{-(\beta_0+\beta_1 c)})^{-1}\mathrm{d}c =$$

Taking $u = \beta_0 + \beta_1 c$ and $\mathrm{d}u = \beta_1 \mathrm{d}c \implies \mathrm{d}u/\beta_1 = \mathrm{d}c$:

$$\frac{1}{\beta_1} \int_0^b (1 + e^{-u})^{-1}\mathrm{d}u =$$

$$\frac{1}{\beta_1} \ln(1 + e^u)\ \Big|_0^b =$$

$$\frac{1}{\beta_1} \ln(1 + e^{(\beta_0+\beta_1 c)})\ \Big|_0^b =$$

$$\frac{1}{\beta_1} \left[\ln(1 + e^{(\beta_0+\beta_1 b)}) - \ln(1 + e^{\beta_0})\right] =$$

$$\frac{1}{\beta_1} \left[\ln \frac{(1 + e^{(\beta_0+\beta_1 b)})}{(1 + e^{\beta_0})}\right] =$$

$$E(cost|bid) = b - \frac{\ln \frac{(1+e^{(\beta_0+\beta_1 b)})}{1+e^{\beta_0}}}{\beta_1 F(b)} =$$

$$b - \frac{\ln \frac{(1+e^{(\beta_0+\beta_1 b)})}{1+e^{\beta_0}} \left(1 + e^{-(\beta_0+\beta_1 b)}\right))}{\beta_1}$$

Where we can get β_0 and β_1 even if we can't easily access to them by following this procedure:

We first select b_1 and b_0, two arbitrary chosen bids, $b_0 \leq b_1$ and get:

$$w_i = -\ln\left(\frac{1}{F_i} - 1\right)$$

w_i is equal to the inner product of the variables and the estimators parameters before applying the sigmoidal "logistic" function from the logistic regression estimator (F_i is the win rate estimation for bid i). Here we get it by applying the logit, the inverse function to the sigmoidal logistic function.

The we can get β_1 doing:

$$\beta_1 = (w_1 - w_0)/(b_F - b_0)$$

This comes from:

$$\frac{(\beta_0 + \beta_1 * b_1) - (\beta_0 + \beta_1 * b_0)}{b_1 - b_0} = \frac{\beta_1 * b_1 - \beta_1 * b_0}{b_1 - b_0} = \frac{\beta_1 * (b_1 - b_0)}{b_1 - b_0} = \beta_1$$

Given that we choose different values for b_0 and b_1, then β_0 can be calculated:

$$\beta_0 = w_0 - \beta_1 b_0$$

3.6.4 Algorithm

Turning the analytical approach into code, we have:

```
def analytical_prediction(estimator, observation, bid0=0.5, bid1=1.0):
    cost_b0, cost_b1 = analytical_first_step(estimator, observation, bid0,
    ↪ bid1)
    if cost_b1 <= 0:
        return 0.0
    bid = observation.bid
    eta0 = cost_b0
    eta1 = cost_b0 + cost_b1 * bid
    a0 = 1 + exp(eta0)
    a1 = 1 + exp(eta1)
    wr0 = 1 / (1 + exp(-eta0))
    wr1 = 1 / (1 + exp(-eta1))
    return bid - (log(a1 / a0) / cost_b1 - wr0 * bid) / (wr1 - wr0)


def analytical_first_step(estimator, observation, bid0, bid1):
    observation.bid = bid0
    eta0 = -log(1 / min(estimator.estimate(observation), 1 - 1e-12) - 1)
    observation.bid = bid1
    eta1 = -log(1 / min(estimator.estimate(observation), 1 - 1e-12) - 1)
    cost_b1 = (eta1 - eta0) / (bid1 - bid0)
    cost_b0 = eta0 - cost_b1 * bid0
    return cost_b0, cost_b1
```

Listing 3.3: Cost estimation using a win rate estimator and doing analytical integration.

Some regards about the implementation are that we take defensive approaches in the logarithms and that we can select any two *bid0* and *bid1* values to calculate *cost_b0* and *cost_b1*, corresponding to β_0 and β_1 in the equations.

This has to be taken into account when considering the cpu performance, as it requires us to perform two estimations.

3.7 Dummy baseline models

In order to have a baseline for the models, we will include the following three estimators:

1. Predicting always the bid floor, as if we were the only bidders participating in the auction.
2. Predicting always our own bid, as if there were a second bidder right below us.
3. Predicting the mean between the bid floor and our own bid.

4. EXPERIMENTS

4.1 Description

As described in the *Appendix II*, we have datasets consisting on samples from two weeks worth of data, and a final day to tests our estimators.

The experiments consists of fitting the estimators described in the Models section, using the algorithms from the Algorithms one, and finally predict the cost of the impressions on the first hours of the last day, as this type of progressive validation is similar to the actual business' requirements, in which we have to estimate an impression's costs in a small time window after fitting an estimator. This type of progressive validation is done on a rolling sliding window of seven days, each fold starting one day after the previous fold. The mean and the standard deviation are shown.

Finally, the evaluation metrics of the first hours of the last day's predictions will be compared.

4.2 Costs dataset exploration

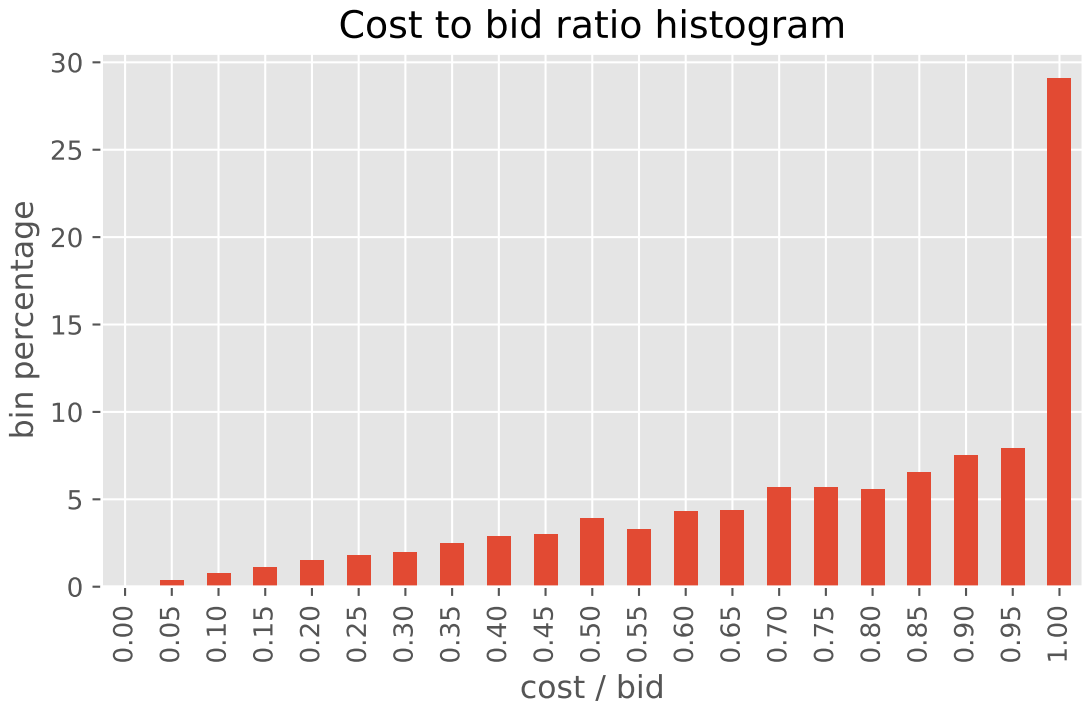

Fig. 4.1: Cost to bid ratio histogram from a sample of the dataset. Each bar shows the percentage of total observations in the bucket. An increasing trend in the percentage of observations in the bucket as the cost ratio increases can be seen.

In figure 4.1 we have a histogram of where impressions' costs falls between the auction's floor and the bid, taken from a sample of the dataset. The most noticeable thing in it, is that the bin corresponding to paying the bid, is considerably larger than the others, accounting for more than 25% of the cases.

This is particularly striking considering that for this to happen, the second highest bidder has to bid right below our bid. This could indicate that different bidders are valuing that impression almost equally (assuming that auction houses are not committing fraud and taking advantage of bidders by reporting the highest price possible).

Looking further into this, we have that in 15% of the auctions, the bid was paid without it being the floor, while in 6% of the cases the floor was bidded and paid.

After further research on this topic, we were unable to find patterns for the cases in which the bid was paid without it being the floor and we will heed no further attention. As a final note, we can comment that DSP often encounter problems of this nature, in which the exchanges seem to be manipulating auction's prices as commented in [45] or including hidden soft floor prices, what means that if the buyer bids between the hard visible floor and the soft floor, the exchange will consider it a first price auction and report the bid as the cost, as explained in [49].

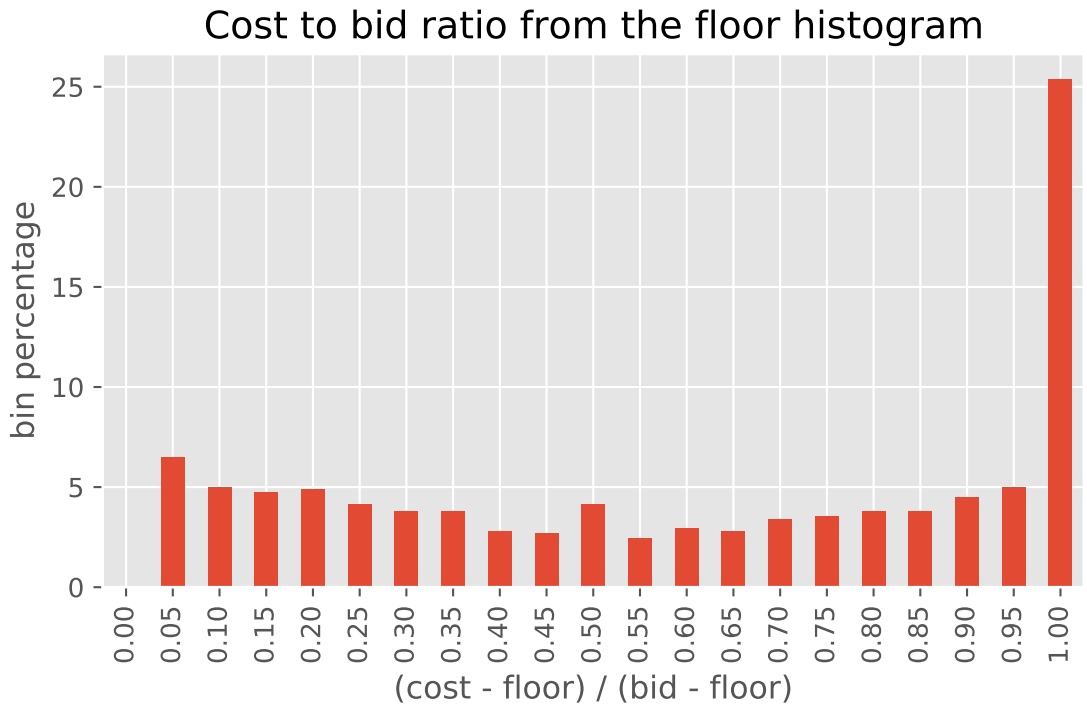

Fig. 4.2: Cost to bid ratio from the floor histogram from a sample of the dataset. Each bar shows the percentage of total observations in the bucket. A spike can be seen when the ratio is 1.0.

In figure 4.2 we have a histogram of the cost to bid ratio from the auction's floor, as in the previous one, it is evident that there are a lot of cases falling on the bin where the cost is equal to the bid, making the histogram hard to associate certainly with a given

distribution, which could mean that the estimation might not work very well.

4.3 Win rate dataset exploration

Observations	6,701,201
Mean	0.92515
Standard deviation	2.84784
Min bid	0.02000
Percentile 25	0.10000
Percentile 50	0.24000
Percentile 75	0.57000
Percentile 80	0.77000
Percentile 90	1.77000
Percentile 93	2.80000
Percentile 95	4.13000
Percentile 99	13.33000
Max bid	425.00000

Tab. 4.1: Bids percentiles taking a sample of a day worth of data

In table 4.1 taken from the win rate dataset, it can be see that most bids are concentrated on small values, having the 75th percentile on 0.57 and the maximum value on 425. Taking this into consideration, we could consider splitting a single estimator into two, one for low bids and one for high bids, as observations with high bids could distort the estimation for low values.

An estimator splitting on low and high bids will be tested against the equivalent single estimator. The value at which the bids will be split will be defined using the same rolling cross-validation scheme as with the rest of the evaluations.

4.4 Win rate estimations exploration

In this section we will present a brief exploration on the win rate estimations, by analyzing the first hour of the following day worth of data. As we would like to know how does the bid impact the observations, estimations and win rate.

The first thing we notice is that there is a long tail of data points with high bids and a small amount of observations, roughly in 98.138% the bids were below $10 and in 99.772% of the cases, the bids where below $20, while in the rest of the cases, the bid prices go up to $425.

By seeing this, we can expect the learning data for high bid prices to be few and to have worse estimations for this cases.

In the following plots, bids higher than 20 have been left out, to avoid ruining the scale of the long tail.

As it can be seen in figure 4.3, most of the bids are concentrated on lower values, which translates in more training data for them and that can imply better estimations for lower bids.

When we see the win rate and the logarithmic loss [14] calculated for observations having the same bid price in figures 4.4 and 4.5, we can see that even though the win rate

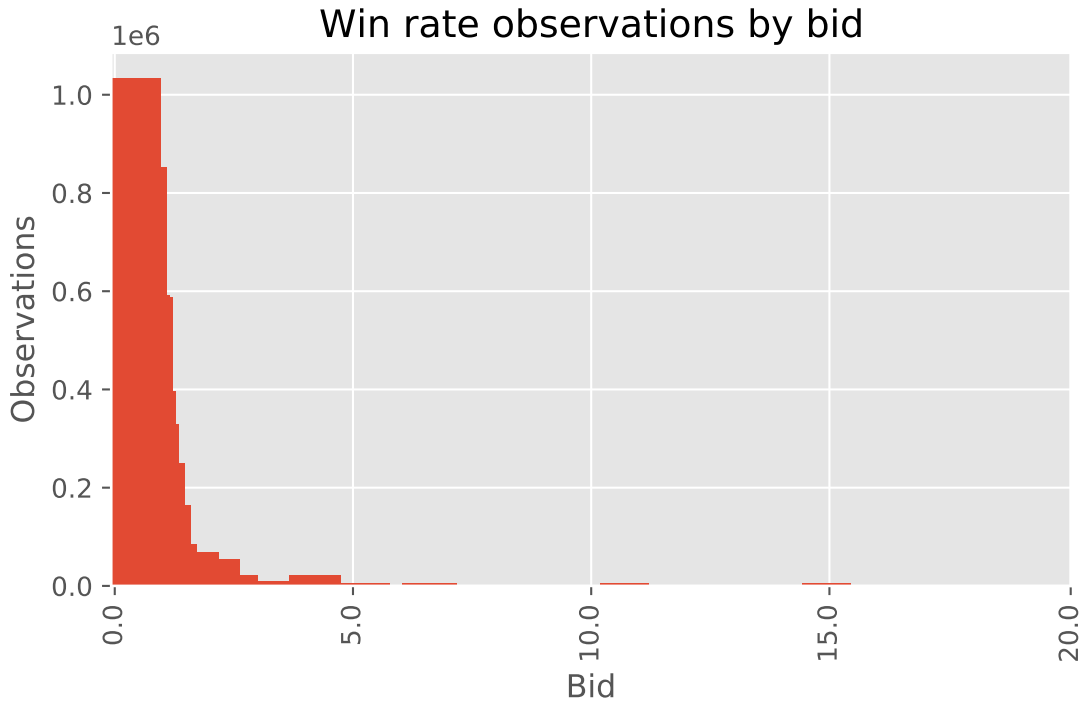

Fig. 4.3: Amount of observations by bid during one hour worth of data (an hour tipycally contains between 15 and 20 million observations). It can be seen that observations are concentrated on lower bids and the amount decreases rapidly. The observations with bids above $20 were left out of the plot (roughly the 0.2% of the observations).

seems to be pretty uniform after a initial growth, on the other hand, the logarithmic loss increases after the $15 bid price, which could mean better cost prediction for lower bids.

One more implication of the uniformity of the win rate depending on the bid price's plot, is that increasing the amount of points for the numeric integration, might not improve the cost estimation.

4.5 Amount of predictions needed

As all the approaches are using the same base model and algorithm, their performance is distinguished in two ways:

1. The amount of data to store and fit.
2. The amount of predictions needed to output a cost prediction.

On this study, we won't get into the cost of storing and fitting more data vs getting better predictions tradeoff that can potentially arise, if the predictions based on the win rate datasets are better.

Each prediction requires a dot product between a feature and the estimator's coefficients vector. In table 4.2 we show the amount of predictions required for each method. It might

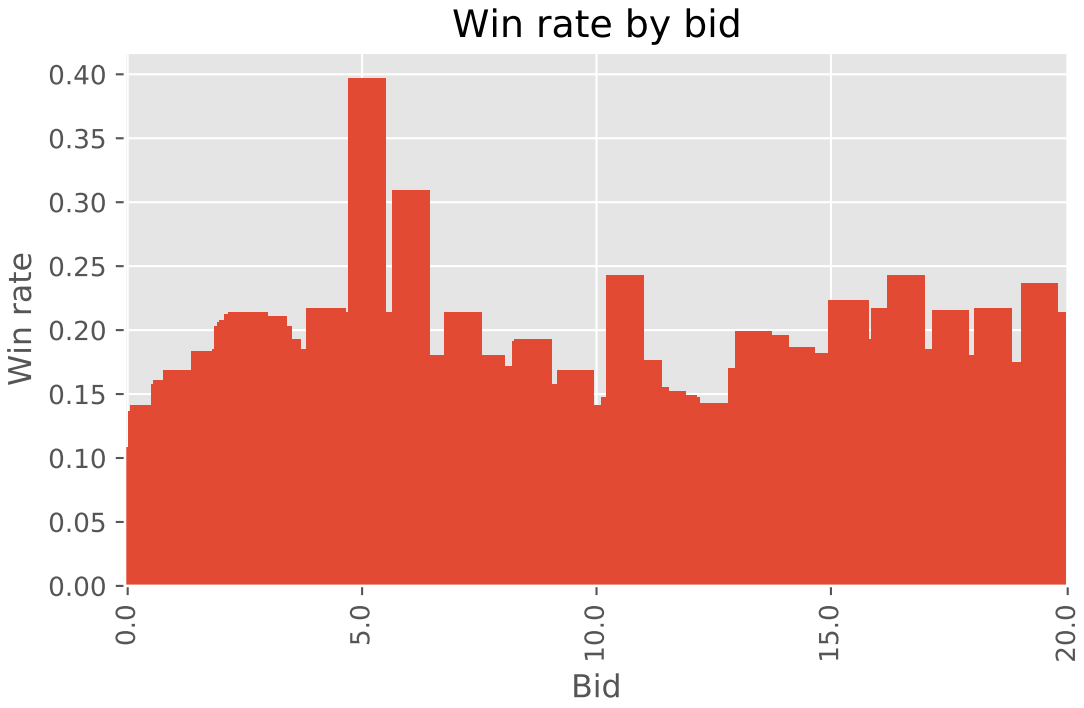

Fig. 4.4: Win rate calculated for observations sharing the same bid during one hour worth of data (an hour tipycally contains between 15 and 20 million observations). Besides some spikes and an initial increase the win rate looks close to uniform. The observations with bids above $20 were left out of the plot (roughly the 0.2% of the observations).

be worth saying that using a Python implementation the estimator takes less than 3 milliseconds to output a single estimation.

4.6 Evaluation metrics

The metrics used to compare the different methods are the **root mean squared prediction error**, **mean absolute error** and the **R squared (coefficient of determination)**.

4.6.1 Root mean squared prediction error

The root mean squared prediction error is always non-negative, a value of 0 indicates a perfect fit and the lower the value, the better.

Given the labels y_i in our dataset, and their corresponding estimations $\hat{y}$, it is defined as: [54]

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

As the errors are squared, the metric is susceptible to outliers as in biggest values and differences, and is harder to interpret.

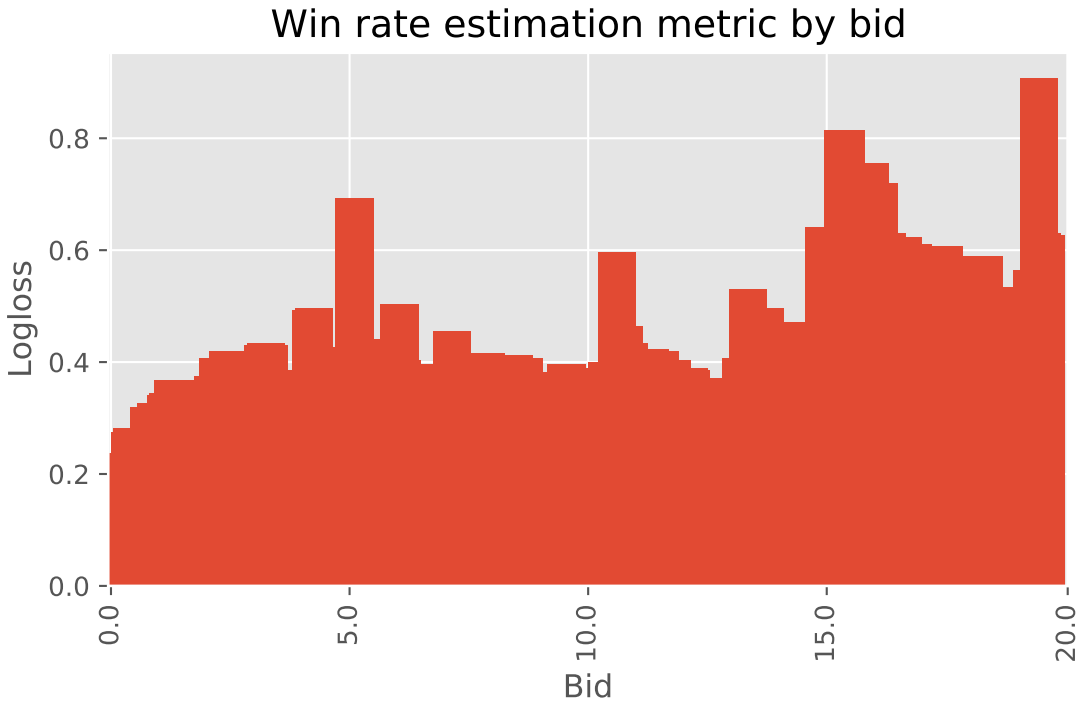

Fig. 4.5: Logloss calculated for observations sharing the same bid using the win rate model for one hour worth of data (an hour tipycally contains between 15 and 20 million observations). The metric shows an increases before $2.5 and after the $15 bid price. The observations with bids above $20 were left out of the plot (roughly the 0.2% of the observations).

4.6.2 Mean absolute error

The mean absolute error (MAE) is a measure of difference between two continuous variables and its value is in the same scale as the dependant variable. Given the labels y_i in our dataset, and their corresponding estimations $\hat{y}$, it is defined as: [53]

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

As this metric shows the average absolute difference between the actual values and the estimations, we can easily interpret by how much we are missing our predictions on average. The lower the MAE, the better.

4.6.3 R^2

The *R squared*, also called coefficient of determination, is a metric often used in regression problems and it is usually described as "the proportion of the variance in the dependent variable that is predictable from the independent variables". In models containing an intercept, the R^2 ranges between 0 and 1, where 1 indicates a perfect fit and lower is worst. [51]

Approach	Predictions needed
Linear model on costs	1 cost prediction
Logistic model	1 cost ratio prediction
Win rate predictions plus numeric integration	At least 2 win rate predictions
Win rate predictions plus analytic integration	2 win rate predictions

Tab. 4.2: Amount of operations required for a single cost prediction, per approach

Given the labels y_i in our dataset, and their corresponding estimations $\hat{y}_i$, R^2 is defined as:

$$R^2 = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

Where:

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

4.6.4 Chosen metrics

The MAE is easy to interpret in this case as it represents the average error in dollars / 1000. It does not indicate if we are under or over predicting, but allows us to compare models by how off they are in monetary terms.

The RMSE weighs more the cases with bigger errors, this makes it harder to interpret, but can tell us if a model has cases in which prediction errors where considerably larger than other model.

The R^2 metric is useful to consider the variance the models account for. It has the advantage to be easily interpretable in the sense that an R^2 below zero tells us that predicting the mean for every case is better than our model and that the closer to 1, the better. Also the more variance the model accounts for, we can expect more precise predictions and smaller error intervals.

If a model has an R^2 lower than zero, we will discard it and work to obtain a better one.

In case two models have contradictory MAE and RMSE metrics (i.e.: one model has better MAE but worse RMSE), we will opt for the model with the best RMSE as we consider better to avoid cases with larger error in the predicted cost, and therefore in the expected net revenue.

For the RMSE and R^2, two models cannot have contradictory RMSE and R^2, if one has a better RMSE, it will also have a R^2.

The Squared Error is defined as $\sum_i (y_i - \hat{y}_i)^2$.

The RMSE is the square root of dividing the squared error by n (the amount of observation). As both are non-decreasing, a higher / lower squared error implies a higher / lower RMSE.

The R^2 for a given test set, is 1 minus the squared error divided by a constant positive number that depends on the dataset. As dividing by a positive number is non-decreasing, an increase / decrease in the square error implies a decrease / increase in the R^2.

This means that for a given test set, if a model has a better Squared Error than other model, it will also have a better RMSE and R^2.

By looking at the R^2 we can determine if a model is better than always predicting the mean value and how much variation it accounts for, with the MAE we can get an idea of the average error in dollars, and with the RMSE we can compare models weighing more cases with larger errors, which we want to avoid.

4.7 Results

Estimator	Capped	RMSE	MAE	R^2
Linear cost	No	1.9624 (0.1404)	0.7273 (0.0475)	0.0563 (0.0058)
Linear cost	Yes	1.6795 (0.1301)	0.3387 (0.0270)	0.3091 (0.0179)

Tab. 4.3: Evaluations metrics of the linear regression method

Estimator	Method	Capped	RMSE	MAE	R^2
Logistic ratio	cost ratio	No	1.0761 (0.1523)	0.1679 (0.0118)	0.7154 (0.0535)
Logistic ratio	cost ratio	Yes	1.0468 (0.1446)	0.1552 (0.0098)	0.7306 (0.0492)
Logistic ratio	floor distance ratio	No	**1.0132 (0.1400)**	**0.1449 (0.0076)**	**0.7460 (0.0566)**
L. R. Split	floor distance ratio	No	**0.9942 (0.1092)**	**0.1426 (0.0080)**	**0.7562 (0.0408)**

Tab. 4.4: The evaluation metrics of the logistic methods

Estimator	Capped	RMSE	MAE	R^2
Win rate analytic	No	1.3514 (0.1433)	0.4087 (0.0344)	0.5529 (0.0445)
Win rate analytic	Yes	**1.0263 (0.1391)**	**0.2413 (0.0226)**	**0.7421 (0.0409)**

Tab. 4.5: Evaluations metrics of the win rate analytic integration method

Estimator	Points	Distribution	Base	Capped	RMSE	MAE	R^2
Win rate numeric	2	uniform	-	No	1.3807 (0.1392)	0.3092 (0.0232)	0.5337 (0.0345)
Win rate numeric	2	uniform	-	Yes	1.3061 (0.1279)	0.2332 (0.0208)	0.5825 (0.0329)
Win rate numeric	10	uniform	-	No	1.4333 (0.1826)	0.3089 (0.0241)	0.4980 (0.0625)
Win rate numeric	10	uniform	-	Yes	1.3572 (0.1718)	0.2324 (0.0216)	0.5497 (0.0577)
Win rate numeric	2	logspace	5	No	1.3800 (0.1391)	0.3146 (0.0232)	0.5342 (0.0344)
Win rate numeric	2	logspace	5	Yes	1.3046 (0.1279)	0.2327 (0.0208)	0.5834 (0.0328)
Win rate numeric	10	logspace	5	No	1.4346 (0.1836)	0.3146 (0.0240)	0.4972 (0.0628)
Win rate numeric	10	logspace	5	Yes	1.3578 (0.1729)	0.2321 (0.0215)	0.5493 (0.0579)
Win rate numeric	2	logspace	10	No	1.3812 (0.1393)	0.3070 (0.0232)	0.5334 (0.0345)
Win rate numeric	2	logspace	10	Yes	1.3068 (0.1279)	0.2336 (0.0208)	0.5820 (0.0329)
Win rate numeric	10	logspace	10	No	1.4358 (0.1832)	0.3072 (0.0239)	0.4963 (0.0626)
Win rate numeric	10	logspace	10	Yes	1.3602 (0.1724)	0.2332 (0.0214)	0.5478 (0.0576)

Tab. 4.6: Evaluations metrics of the win rate numeric integration method

Estimator	RMSE	MAE	R^2
Dummy: floor	1.8026 (0.1323)	0.3951 (0.0341)	0.2038 (0.0140)
Dummy: bid	2.0650 (0.2018)	0.3727 (0.0266)	-0.0448 (0.0931)
Dummy: (floor + bid) / 2	**1.1702 (0.1021)**	**0.2708 (0.0198)**	**0.6649 (0.0143)**

Tab. 4.7: Evaluations metrics of the dummy methods

Estimator	RMSE	MAE	R^2
Linear cost	1.6795 (0.1301)	0.3387 (0.0270)	0.3091 (0.0179)
Logistic ratio	1.0468 (0.1446)	0.1552 (0.0098)	0.7306 (0.0492)
Logistic ratio from the floor	**1.0132 (0.1400)**	**0.1449 (0.0076)**	**0.7460 (0.0566)**
Win rate analytic	1.0263 (0.1391)	0.2413 (0.0226)	0.7421 (0.0409)
Win rate numeric	1.3046 (0.1279)	0.2327 (0.0208)	0.5834 (0.0328)
Dummy: (floor + bid) / 2	1.1702 (0.1021)	0.2708 (0.0198)	0.6649 (0.0143)

Tab. 4.8: Evaluations metrics among methods. Selecting one of each type.

4.8 Discussion

In all cases in the first four tables, the Capped estimators (setting the final output between the floor and the bid) obtained better metrics, which indicates that the models as they are, fail to capture that the cost has to be greater than or equal to the floor, and lower or equal to the bid (exceptuating the *cost ratio from the floor* model).

In table 4.4, comparing the results of the logistic based approaches, we can see that the second method, estimating the ratio from the floor, is overall a better estimator than the whole ratio. On this table, it can also be seen that even though using two estimators, splitting the cases depending on the bid value (< 3 on the shown results), can result in a small improvement of the metrics, even though we consider it is not worth the extra effort.

In table 4.5, the improvement of the win rate with the analytical integration method considering the cap is noticeable improving almost a 37% on the R^2.

In table 4.6, as in table 4.5, there is an improvement capping the output, although it is not as big as for the analytical method, which could mean that the analytical method more often outputs predictions below the floor and is beneficiated by including the cap, and at the same time is better estimating the costs that are above the floor. Being consistent with what we expected from the win rate by bid analysis, increasing the amount of points for the numeric integration does not improve the estimation, what is worst, it deteriorates them. This also impacts in the fact, that as fewer points are better, how we distribute the points does not enhance the predictions either.

In table 4.7, we show the results of three dummy approaches trying to guess the cost using only the floor and the bid, with pre-defined rules. We can see that always answering the floor or the bid, does not yield good results, yet, always yielding halfway between them gets decent results and we will use this estimator as the baseline.

In table 4.8 we summarized the best estimator from each type. From the go we can see that the linear regression and the win rate with numeric integration perform a poor job predicting the job, getting worse metrics than the dummy approach of using the mean between the floor and the bid as the estimated cost. Between the first two, it can be seen that the latter obtained better results than the former, the same way as reported in [26].

The capped logistic ratio between 0 and the bid, the logistic ratio between the floor and the bid, and the capped win rate with analytic integration methods outperform the dummy approach, while the increase in R^2 is of 9%+, the logistic methods obtain a MAE more than 44% lower than the dummy's one.

Comparing the win rate with analytic integration method against the logistic based ones, although the difference between the RMSE and the R^2 are lower than a 4%, the MAE makes the difference by more than a 37% decrease on the logistic methods. This could be interpreted as that the errors made by the logistic methods are generally smaller but the difference between the biggest errors made by the logistic ones are similar to the analytic methods or even slightly larger.

Finally, among the logistic methods, the one that predicts the ratio considering between the auction's floor and the bid, got overall better results, mostly relevant in the MAE, having a difference close to a 6%.

We consider the logistic based approach estimating the cost ratio starting from the floor to the bid, as the best estimator, even though modelling the problem this way, takes extra steps, as adding/substracting the floor to the estimations / labels, it captures the output's valid range and it is not needed to limit results that could be outside of the valid values.

Based on the results, we can tell that taking into consideration knowledge of the problem's domain and how this affects the modelling and possible outputs is of great importance. As an example, just by limiting the output to the possible values, the results were significantly improved for naive statistical learning approaches. This also highlights that statistical learning and machine learning should not be considered as a simple out of the box cure-all strategy and that considering other approaches or enhacing the solutions with domain knowledge can vastly improve the results, which was also seen by the dummy approach outperforming the linear regression, which is most likely the first thing a machine learning practitioner would try.

5. CONCLUSION

In this work we compared different approaches for predicting the cost of second price auctions in a real time bidding environment.

Stochastic online gradient descent optimized linear and logistic regression algorithms along with the hashing trick were used to fit a dataset with million of data points corresponding to two weeks worth of time changing data from a DSP.

Previously studied methods, i.e. a linear regression on the cost of won auctions and numerically integrating a logistic regression on the probability of winning an auction, were compared against new approaches, i.e. a logistic regression on the ratio of the cost to the bid price, a logistic regression on the ratio of the cost to the bid price starting from the auction's bid price floor, and analytically integrating a logistic regression on the probability of winning an auction.

The results obtained show that the proposed logistic regression based approached outperform the other approaches and that the analytical integration outperforms the numerical integration.

This could allow for better cost predictions on second price auctions, which can be used to improve the bidding strategies of DSPs, which could help improve their net revenue and their clients' return of investment. Using the logistic based approached would also allow to use smaller datasets, which can result in faster fitting and hyperparameter optimization, and cheaper maintenance as less data would have to be stored and processed, which can have an impact on cloud based solutions.

6. FUTURE WORK

As future work, we want to make a deeper analysis of the tradeoff between the amount of data required to fit the win rate estimations and amount of predictions needed vs the performance improvement of having better cost estimations.

Another research opportunity we want to explore is using other algorithms to fit the models, as random forests in a single computer, random forests, gradient boosting trees and logistic regression using distributed frameworks as Spark's MLlib, and deep learning approaches. All of these require us to evaluate the cost of using them vs the potential performance improvement from switching to them. As they all are more expensive to run as they require bigger or more servers than the logistic estimator used in this work:

- Random forests in a single computer would require a large amount of RAM memory to load and fit the estimator, with the potential requirement of fitting the data in batches if there is not enough RAM.
- Spark's MLlib distributed algorithms need more than one worker in order to keep up with a single machine implementation, to cover for the node communication costs, which requires to pay for several servers and data transfers costs.
- Deep learning requires GPUs in order to achieve fast learning times, which translates to more expensive servers.

One final point we want to venture into, is to research using an estimator from bids to costs, in which we consider all the auctions. In this case, lost auctions would have a cost of zero, and as we see it, the advantage of this approach, would be that we would consider both win rate and cost, all at once, while as it was proposed in this work, estimating the cost given that we win the auction, we are potentially losing information as we are ignoring the fact that there could be a certain covariance among both of them that we are not considering in the expected net revenue.

BIBLIOGRAPHY

[1] Hirotugu Akaike. "A new look at the statistical model identification". In: *IEEE transactions on automatic control* 19.6 (1974), pp. 716–723.

[2] Cristián Antuna et al. "Una plataforma de Real Time Bidding escalable e inteligente". In: *Simposio Argentino de GRANdes DAtos (AGRANDA 2015)-JAIIO 44 (Rosario, 2015)*. 2015.

[3] Apsalar. *Take 5 – What are CPM, CPC, CPL, CPA, CPS and CPI Media Buying Models?* https://apsalar.com/wp-content/uploads/2015/08/Media-Buying-Models_NewCover.pdf. [Online; accessed 24-November-2019]. Apsalar, 2015.

[4] Avazu Inc. *Avazu homepage*. http://avazuinc.com/home/. [Online; accessed 02-March-2019]. Avazu Inc., 2019.

[5] AWS Customer Success. *Big Data Pays Off Big Time for Jampp on AWS*. https://aws.amazon.com/solutions/case-studies/jampp/. [Online; accessed 02-March-2019]. Adexchanger, 2017.

[6] Christoph Bergmeir and José M Benítez. "On the use of cross-validation for time series predictor evaluation". In: *Information Sciences* 191 (2012), pp. 192–213.

[7] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[8] Léon Bottou, Frank E Curtis, and Jorge Nocedal. "Optimization methods for large-scale machine learning". In: *Siam Review* 60.2 (2018), pp. 223–311.

[9] Robert S Boyer and J Strother Moore. "MJRTY—a fast majority vote algorithm". In: *Automated Reasoning*. Springer, 1991, pp. 105–117.

[10] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. "Simple and scalable response prediction for display advertising". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 5.4 (2015), p. 61.

[11] Junxuan Chen et al. "Deep ctr prediction in display advertising". In: *Proceedings of the 24th ACM international conference on Multimedia*. ACM. 2016, pp. 811–820.

[12] Criteolabs. *It's All About "Big Data"*. https://labs.criteo.com/2017/05/its-all-about-big-data/. [Online; accessed 20-March-2019]. Criteo, 2017.

[13] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. 2nd ed. New York: Wiley, 2001. ISBN: 978-0-471-05669-0.

[14] Fast AI wiki contributors. *Log loss*. http://wiki.fast.ai/index.php/Log_Loss. [Online; accessed 25-November-2018]. 2017.

[15] Google. *Latency Restrictions and Peering — Real-Time Bidding*. https://developers.google.com/authorized-buyers/rtb/peer-guide. [Online; accessed 24-November-2019]. Google, 2019.

[16] *Google Play Store statistics*. https://42matters.com/stats. [Online; accessed 25-November-2018]. 42 matters, 2018.

[17] Thore Graepel et al. "Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine". In: Omnipress. 2010.

[18] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[19] Xinran He et al. "Practical lessons from predicting clicks on ads at facebook". In: *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. ACM. 2014, pp. 1–9.

[20] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[21] Yuchin Juan et al. "Field-aware factorization machines for CTR prediction". In: *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM. 2016, pp. 43–50.

[22] Paul Klemperer. "Auction theory: A guide to the literature". In: *Journal of economic surveys* 13.3 (1999), pp. 227–286.

[23] Ron Kohavi. "A study of cross-validation and bootstrap for accuracy estimation and model selection". In: *Proceedings of the 14th international joint conference on Artificial intelligence-Volume 2*. Morgan Kaufmann Publishers Inc. 1995, pp. 1137–1143.

[24] Langford, J., Li, L., Strehl, A. *Vowpal wabbit online learning project (Technical Report): Feature hashing in vowpal wabbit*. https://github.com/VowpalWabbit/vowpal_wabbit/wiki/Feature-Hashing-and-Extraction. [Online; accessed 25-November-2018]. 2007.

[25] Ron Lavi. "Algorithmic game theory". In: *Computationally-efficient approximate mechanisms* (2007), pp. 301–330.

[26] Xiang Li and Devin Guan. "Programmatic buying bidding strategies with win rate and winning price estimation in real time mobile advertising". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2014, pp. 447–460.

[27] Xiaoliang Ling et al. "Model ensemble for click prediction in bing search ads". In: *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee. 2017, pp. 689–698.

[28] David Lucking-Reiley. "Vickrey auctions in practice: From nineteenth-century philately to twenty-first-century e-commerce". In: *Journal of economic perspectives* 14.3 (2000), pp. 183–192.

[29] Brendan McMahan. "Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011, pp. 525–533.

[30] H Brendan McMahan et al. "Ad click prediction: a view from the trenches". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2013, pp. 1222–1230.

[31] Paul Milgrom and Paul Robert Milgrom. *Putting auction theory to work*. Cambridge University Press, 2004.

[32] Mobfox. *Integrate as a DSP - Mobfox SSP Documentation - Confluence*. https://mobfox.atlassian.net/wiki/spaces/PUMD/pages/466518062/Integrate+as+a+DSP. [Online; accessed 24-November-2019]. Mobfox, 2019.

[33] Mopub. *Network Infrastructure — MoPub OpenRTB Integration — MoPub Developers*. https://developers.mopub.com/dsps/integration/network-infrastructure/. [Online; accessed 24-November-2019]. Mopub, 2019.

[34] Roger B Myerson. "Optimal auction design". In: *Mathematics of operations research* 6.1 (1981), pp. 58–73.

[35] OpenX. *BidResponse object*. https://docs.openx.com/Content/demandpartners/openrtb_bidresponse.html. [Online; accessed 24-November-2019]. OpenX, 2019.

[36] Carlos Pita. "Sampling RTB transactions in an online machine learning setting". In: *II Simposio Argentino de GRANdes DAtos (AGRANDA 2016)-JAIIO 45 (Tres de Febrero, 2016)*. 2016.

[37] Suju Rajan. *Computational Advertising at Scale*. KDD, 2018.

[38] Z Reitermanova. "Data splitting". In: vol. 10. 2010.

[39] Matthew Richardson, Ewa Dominowska, and Robert Ragno. "Predicting clicks: estimating the click-through rate for new ads". In: *Proceedings of the 16th international conference on World Wide Web*. ACM. 2007, pp. 521–530.

[40] John Riley and William Samuelson. "Optimal auctions". In: *The American Economic Review* 71.3 (1981), pp. 381–392.

[41] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2016. arXiv: 1609.04747 [cs.LG].

[42] Gideon Schwarz et al. "Estimating the dimension of a model". In: *The annals of statistics* 6.2 (1978), pp. 461–464.

[43] Cosma Rohilla Shalizi. *Advanced Data Analysis from an Elementary Point of View*. Cambridge University Press, 2018. Chap. 3.

[44] John Shawe-Taylor et al. "Structural risk minimization over data-dependent hierarchies". In: *IEEE transactions on Information Theory* 44.5 (1998), pp. 1926–1940.

[45] Sluis, Sarah. *MediaMath Will Drop Supply Partners That Game The Auction*. https://adexchanger.com/platforms/mediamath-will-drop-supply-partners-that-game-the-auction/. [Online; accessed 25-November-2018]. Adexchanger, 2018.

[46] Leonard J Tashman. "Out-of-sample tests of forecasting accuracy: an analysis and review". In: *International journal of forecasting* 16.4 (2000), pp. 437–450.

[47] Tim Roughgarden. *CS 269I: Incentives in Computer Science. Auction basics*. http://timroughgarden.org/f16/f16.html. [Online; accessed 23-March-2019]. 2016.

[48] William Vickrey. "Counterspeculation, auctions, and competitive sealed tenders". In: *The Journal of finance* 16.1 (1961), pp. 8–37.

[49] Weatherman, Kevin. *Introducing Soft Price Floors*. https://www.mopub.com/2013/04/12/introducing-soft-price-floors. [Online; accessed 25-November-2018]. Mopub, 2013.

[50] Kilian Weinberger et al. "Feature hashing for large scale multitask learning". In: *arXiv preprint arXiv:0902.2206* (2009).

[51] Wikipedia contributors. *Coefficient of determination — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Coefficient_of_determination&oldid=870225578. [Online; accessed 25-November-2018]. 2018.

[52] Wikipedia contributors. *Gradient descent — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Gradient_descent&oldid=868455273. [Online; accessed 25-November-2018]. 2018.

[53] Wikipedia contributors. *Mean absolute error — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Mean_absolute_error&oldid=859130031. [Online; accessed 25-November-2018]. 2018.

[54] Wikipedia contributors. *Root-mean-square deviation — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Root-mean-square_deviation&oldid=856934073. [Online; accessed 25-November-2018]. 2018.

[55] Wikipedia contributors. *Trapezoidal rule — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Trapezoidal_rule&oldid=863398468. [Online; accessed 25-November-2018]. 2018.

[56] Wush Chi-Hsuan Wu, Mi-Yen Yeh, and Ming-Syan Chen. "Predicting winning price in real time bidding with censored data". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2015, pp. 1305–1314.

[57] Hongxia Yang et al. "Large Scale CVR Prediction through Dynamic Transfer Learning of Global and Local Features". In: *Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*. 2016, pp. 103–119.

[58] Weinan Zhang et al. "Bid-aware gradient descent for unbiased learning with censored data in display advertising". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 665–674.

[59] Yongli Zhang and Yuhong Yang. "Cross-validation for selecting a model selection procedure". In: *Journal of Econometrics* 187.1 (2015), pp. 95–112.

[60] Yuyu Zhang et al. "Sequential click prediction for sponsored search with recurrent neural networks". In: *Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.

6. APPENDIX I: REAL TIME BIDDING AND AD TECH GLOSSARY

Ad Exchange
1. A virtual marketplace where participating suppliers auction their impressions opportunities to eligible buyers. The ad exchange announces each impressions, in real time, and asks buyers if they are interested to buy said impression and at which price. 2. A technology platform used by digital media buyers and sellers to facilitate automated, auction-based pricing and buying in real-time that provides advertisers with aggregate publisher inventory. Sits between publishers / supply side platforms (SSPs) and advertisers / demand side platforms (DSPs).

Ad impression
A single ad that appears when the opportunity arrives on the viewer's display. A web page or app may offer space for a number of ad views.

Ad tech
Advertising technology. Commonly refers to the technologies that enable the automated or programmatic buying and selling of digital advertising. Often confused with MarTech (marketing technology), which encompasses the tools used to manage marketing processes, workflows, digital content and customer analytics.

Ad
For web advertising, an ad is almost always a banner, graphic image, or set of animated images (in an animated GIF) of a designated pixel size and byte size limit. An ad or set of ads for a campaign is often referred to as "the creative." Banners and other special advertising that include an interactive or visual element beyond the usual are known as rich media ads.

Advertiser
The person or organization interested on the advertisement impression opportunity for showing their ad.

Auction
A call from a publisher / ad exchange making its inventory available to multiple buyers for pricing offers; includes various data points such as location, device, etc. that allow potential buyers to price the impression.

Bid request
An auction

Click
According to the ad industry recommended guidelines from FAST, a click is "when a visitor interacts with an advertisement." This does not mean simply interacting with a rich media ad, but actually clicking on it so that the visitor is headed toward the advertiser's destination. (It also does not mean that the visitor actually waits to fully arrive at the

destination, but just that the visitor started going there.)

Click through Rate (CTR)
The rate (expressed in a percentage) at which users click on an ad. This is calculated by dividing the total number of clicks by the total number of ad impressions.

CVR
Conversion rate. Ratio of actions of interest performed by the users over the clicks on impressions.

Cost
Amount of money paid for an ad impression.

Contextual data
Data related to the content and context of the specific media where advertisement is shown.

CPM
Acronym for "cost per thousand" ad impressions, an industry standard measure for selling ads on websites. This measure is taken from print advertising. The "M" is taken from the Roman numeral for "thousand."

CPx pricing
Refers to how media is bought on a cost per basis. The **x** is replaced by **M** (CPM) to refer to Cost Per Thousand, or **C** (CPC) to refer to Cost Per Click, or any variant of **A** (CPA) Cost Per Action.

Demand Side Platform (DSP)
A DSP is a technology platform through which buyers (Advertisers or Agencies) can plan, target, execute, optimize, and analyze digital media buying programs across 100% of the media plan. Through a DSP, the buyer can set targeting criteria, pricing, frequency, and other criteria governing the purchase of digital ad units. The DSP handles automated media buying for advertisers using unified targeting, data, and RTB optimization via a bidding algorithm.

Impression
Ad impression. When an ad is shown.

OpenRTB
A standard for the Real Time Bidding interface intended to set the requirements bar and simplify the connection between suppliers of publisher inventory (i.e.: ad exchanges, SSPs, etc.) and competing buyers (i.e.: bidders, DSPs, etc.).

Programmatic advertising
The use of software to improve the buying and selling of digital media through workflow automation and algorithms. Innovations in this area have redirected human involvement to more strategic tasks and replaced some repetitive actions with more efficient and effective technologies to drive better targeting and campaign placement optimization.

Publisher

Sites or apps that sell their ad impression spaces.

Real Time Bidding (RTB)
Way of transacting media that allows an individual ad impression to be put up for bid in real time. This is done through a programmatic on the spot auction, which is similar to how financial markets operate. RTB allows for Addressable Advertising; the ability to serve ads (digital media display, video, mobile, social) to consumers directly based on their demographic, psychographic, or behavioral attributes, at the impression level. The auctions are facilitated by ad exchanges as intermediaries between publishers / SSPs and bidders / DSPs.

Real Time Bidding Bidder
A system that connects to one or more ad exchanges and evaluates every impression opportunity that's announced. The real-time bidder is responsible for making the best inventory acquisition decisions possible, by bidding on the auctions on behalf of the Advertisers.

Second Price Auction
Sealed auction in which the winner of the bid pays the price of the 2nd highest bidder (also known as a Vickry auction).

SSP
A technology platform that allows publishers to connect their inventory to multiple demand sources with brand control, pricing, packaging and reporting tools; one component of a holistic inventory management solution for publishers that encompasses all screens, format and channels.

Win Rate
The number of impressions won over the number of impressions bid.

This glossary was assembled from different sources as:

- A DSP's internal documentation.
- Pubmatic's Programmatic On Deck.
- IAB's glossary
- Canada's IAB glossary
- Canada' IAB RTB glossary

6. APPENDIX II: DATASETS

The datasets used consist on a sample of 14 days worth of data from the DSP. The datasets are kept as one compressed csv file per dataset type, per day.

One last day is provided as test set for the costs dataset to perform progressive validation, as estimating future costs in a small time window is what is required from the business' point of view.

Categorical variables have been hashed in order to keep them anonymized.

6.1 Predictor variables

- bid: the bid we offered for the auction's ad space in dollars.
- bid_floor: the auction's minimum acceptable bid in dollars.
- dow: the day of the week in the user's timezone.
- hod: the hour of the day in the user's timezone.
- height: the ad space height in pixels if it corresponds (some ads do not have a predefined size).
- width: the ad space height in pixels if it corresponds (some ads do not have a predefined size).
- categorical_0: a categorical variable from the auction.
- categorical_1: a categorical variable from the auction.
- categorical_2: a categorical variable from the auction.
- categorical_3: a categorical variable from the auction.
- categorical_4: a categorical variable from the auction.
- categorical_5: a categorical variable from the auction.
- categorical_6: a categorical variable from the auction.
- categorical_7: a categorical variable from the auction.

6.2 Responses

- label: 1 in case we won the auction and the ad was shown, 0 otherwise. (For the win rate dataset).
- cost: the auction's ad space cost (in case we won and the was shown). (For the costs dataset).

6. APPENDIX III: EXPECTATION LEMMA PROOF

In 3.6.1 the following lemma is used:

Lemma. If X is a continuous random variable and X is non-negative, then:

$$E(X) = \int_0^\infty P(X > y)\ dy = \int_0^\infty 1 - F_X(y) dy$$

Lemma proof:

For any two numbers x and y, we define:

$$I(y < x) = \begin{cases} 1 & \text{if } y < x \\ 0 & \text{if } y \geq x \end{cases}$$

Then:

$$E(X) = \int_{-\infty}^\infty x\ f_X(x)\ dx$$

And since X is non-negative, $f_X(x) = 0\ \forall\ x \leq 0$:

$$= \int_0^\infty x\ f_X(x)\ dx$$

$$= \int_0^\infty \left(\int_0^\infty I(y < x) dy \right) f_X(x)\ dx$$

$$= \int_0^\infty \int_0^\infty I(y < x)\ f_X(x)\ dy\ dx$$

$$= \int_0^\infty \int_0^\infty I(y < x)\ f_X(x)\ dx\ dy$$

$$= \int_0^\infty \left(\int_0^y [0\ f_X(x)]\ dx \int_y^\infty [1\ f_X(x)]\ dx \right) dy$$

$$= \int_0^\infty \left(\int_y^\infty f_X(x)\ dx \right) dy$$

$$= \int_0^\infty P(X > y)\ dy$$

$$= \int_0^\infty 1 - F_X(y)\ dy$$

Reaching the used result.

6. APPENDIX IV: OPENRTB OBJECT MODEL

In this section we will include the OpenRTB 2.5 object model and variables description. This is the data that is potentially available to use for predicting the second price auctions cost.

3. Bid Request Specification

RTB transactions are initiated when an exchange or other supply source sends a bid request to a bidder. The bid request consists of the top-level bid request object, at least one impression object, and may optionally include additional objects providing impression context.

3.1 Object Model

Following is the object model for the bid request. The top-level object (i.e., in JSON the unnamed outer object) is denoted as `BidRequest` in the model. Of its direct subordinates, only `Imp` is technically required since it is fundamental to describing the impression being sold and its requires at least one of `Banner` (which may allow multiple formats), `Video`, `Audio`, and `Native` to define the type of impression (i.e., whichever one or more the publisher is willing to accept; although a bid will be for exactly one of those specified). An impression can optionally be subject to a private marketplace.

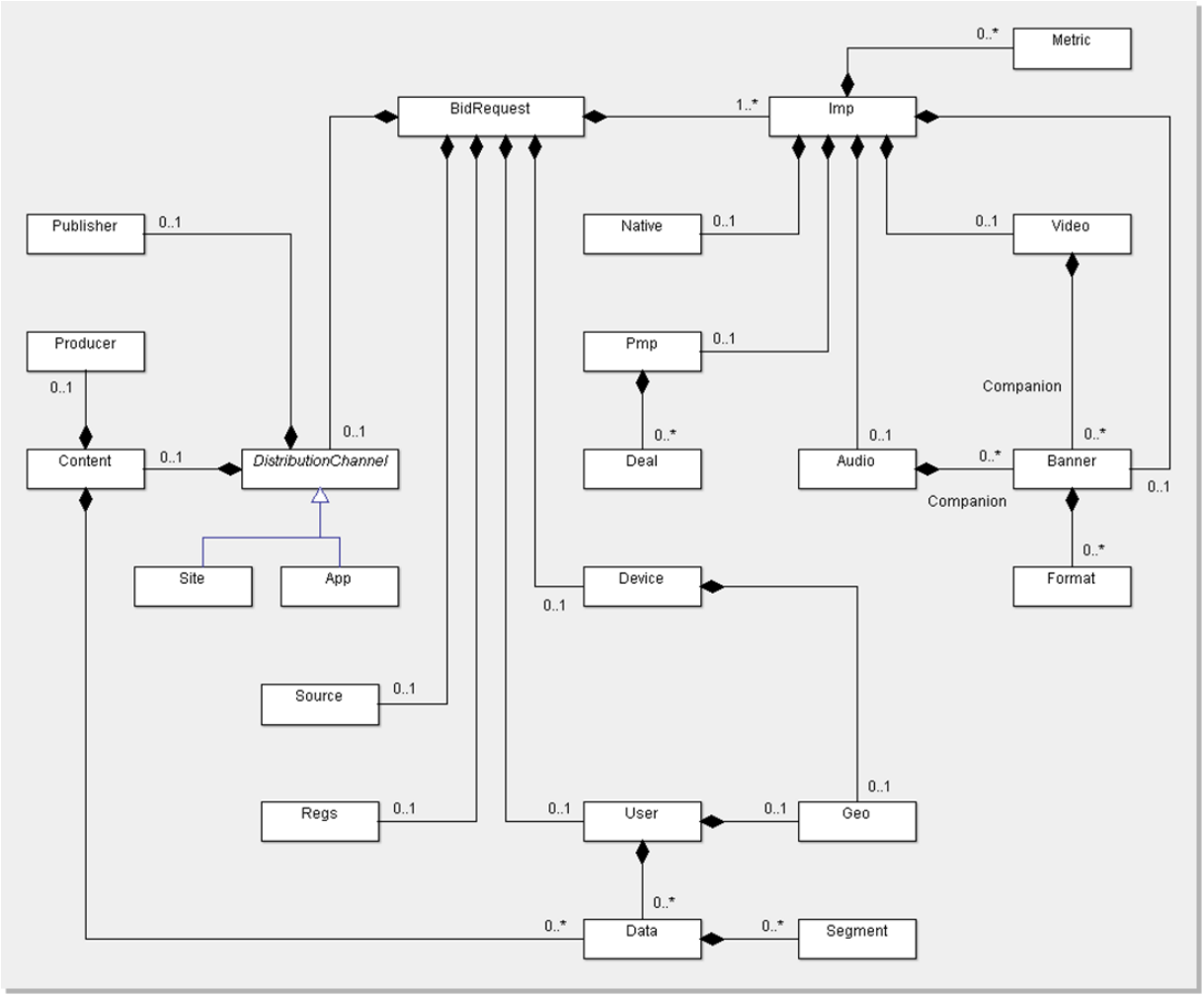

Figure 3: Bid Request object model.

Other subordinates to the `BidRequest` provide various forms of information to assist bidders in making targeting and pricing decisions. This includes details about the user, the device they're using, the location of either, regulatory constraints, and the content and media in which the impression will occur.

On the latter, there is the distinction between site (i.e., website) and application (i.e., non-browser app typically in mobile). The abstract class called `DistributionChannel` is just a modeling concept to indicate that a `BidRequest` is related to either a `Site` or an `App`, but not both (i.e., a distribution channel is an abstraction of site and app). Both sites and apps can be further described by data about their publisher, the content, and the content's producer.

Not shown in the model figure is an extensions object. This is an object of undefined structure that can be added to any other object to convey exchange-specific extensions to the standard. Exchanges using these objects are responsible for publishing their extensions to their bidders.

The following table summarizes the objects in the Bid Request model and serves as an index into the detailed definitions in the subsections that follow.

Object	Section	Description
`BidRequest`	3.2.1	Top-level object.
`Source`	3.2.2	Request source details on post-auction decisioning (e.g., header bidding).
`Regs`	3.2.3	Regulatory conditions in effect for all impressions in this bid request.
`Imp`	3.2.4	Container for the description of a specific impression; at least 1 per request.
`Metric`	3.2.5	A quantifiable often historical data point about an impression.
`Banner`	3.2.6	Details for a banner impression (incl. in-banner video) or video companion ad.
`Video`	3.2.7	Details for a video impression.
`Audio`	3.2.8	Container for an audio impression.
`Native`	3.2.9	Container for a native impression conforming to the Dynamic Native Ads API.
`Format`	3.2.10	An allowed size of a banner.
`Pmp`	3.2.11	Collection of private marketplace (PMP) deals applicable to this impression.
`Deal`	3.2.12	Deal terms pertaining to this impression between a seller and buyer.
`Site`	3.2.13	Details of the website calling for the impression.
`App`	3.2.14	Details of the application calling for the impression.
`Publisher`	3.2.15	Entity that controls the content of and distributes the site or app.
`Content`	3.2.16	Details about the published content itself, within which the ad will be shown.
`Producer`	3.2.17	Producer of the content; not necessarily the publisher (e.g., syndication).
`Device`	3.2.18	Details of the device on which the content and impressions are displayed.
`Geo`	3.2.19	Location of the device or user's home base depending on the parent object.
`User`	3.2.20	Human user of the device; audience for advertising.
`Data`	3.2.21	Collection of additional user targeting data from a specific data source.
`Segment`	3.2.22	Specific data point about a user from a specific data source.

3.2 Object Specifications

The subsections that follow define each of the objects in the bid request model. Several conventions are used throughout:

- Attributes are "required" if their omission would technically break the protocol.
- Some optional attributes are denoted "recommended" due to their elevated business importance.
- Unless a default value is explicitly specified, an omitted attribute is interpreted as "unknown".

3.2.1 Object: BidRequest

The top-level bid request object contains a globally unique bid request or auction ID. This `id` attribute is required as is at least one impression object (Section 3.2.4). Other attributes in this top-level object establish rules and restrictions that apply to all impressions being offered.

There are also several subordinate objects that provide detailed data to potential buyers. Among these are the `Site` and `App` objects, which describe the type of published media in which the impression(s) appear. These objects are highly recommended, but only one applies to a given bid request depending on whether the media is browser-based web content or a non-browser application, respectively.

Attribute	Type	Description
`id`	string; required	Unique ID of the bid request, provided by the exchange.
`imp`	object array; required	Array of `Imp` objects (Section 3.2.4) representing the impressions offered. At least 1 `Imp` object is required.
`site`	object; recommended	Details via a `Site` object (Section 3.2.13) about the publisher's website. Only applicable and recommended for websites.
`app`	object; recommended	Details via an `App` object (Section 3.2.14) about the publisher's app (i.e., non-browser applications). Only applicable and recommended for apps.
`device`	object; recommended	Details via a `Device` object (Section 3.2.18) about the user's device to which the impression will be delivered.
`user`	object; recommended	Details via a `User` object (Section 3.2.20) about the human user of the device; the advertising audience.
`test`	integer; default 0	Indicator of test mode in which auctions are not billable, where 0 = live mode, 1 = test mode.
`at`	integer; default 2	Auction type, where 1 = First Price, 2 = Second Price Plus. Exchange-specific auction types can be defined using values greater than 500.
`tmax`	integer	Maximum time in milliseconds the exchange allows for bids to be received including Internet latency to avoid timeout. This value supersedes any *a priori* guidance from the exchange.
`wseat`	string array	White list of buyer seats (e.g., advertisers, agencies) allowed to bid on this impression. IDs of seats and knowledge of the buyer's customers to which they refer must be coordinated between bidders and the exchange *a priori*. At most, only one of `wseat` and `bseat` should be used in the same request. Omission of both implies no seat restrictions.

bseat	string array	Block list of buyer seats (e.g., advertisers, agencies) restricted from bidding on this impression. IDs of seats and knowledge of the buyer's customers to which they refer must be coordinated between bidders and the exchange *a priori*. At most, only one of wseat and bseat should be used in the same request. Omission of both implies no seat restrictions.
allimps	integer; default 0	Flag to indicate if Exchange can verify that the impressions offered represent all of the impressions available in context (e.g., all on the web page, all video spots such as pre/mid/post roll) to support road-blocking. 0 = no or unknown, 1 = yes, the impressions offered represent all that are available.
cur	string array	Array of allowed currencies for bids on this bid request using ISO-4217 alpha codes. Recommended only if the exchange accepts multiple currencies.
wlang	string array	White list of languages for creatives using ISO-639-1-alpha-2. Omission implies no specific restrictions, but buyers would be advised to consider language attribute in the Device and/or Content objects if available.
bcat	string array	Blocked advertiser categories using the IAB content categories. Refer to List 5.1.
badv	string array	Block list of advertisers by their domains (e.g., "ford.com").
bapp	string array	Block list of applications by their platform-specific exchange-independent application identifiers. On Android, these should be bundle or package names (e.g., com.foo.mygame). On iOS, these are numeric IDs.
source	object	A Sorce object (Section 3.2.2) that provides data about the inventory source and which entity makes the final decision.
regs	object	A Regs object (Section 3.2.3) that specifies any industry, legal, or governmental regulations in force for this request.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.2 Object: Source

This object describes the nature and behavior of the entity that is the source of the bid request upstream from the exchange. The primary purpose of this object is to define post-auction or upstream decisioning when the exchange itself does not control the final decision. A common example of this is header bidding, but it can also apply to upstream server entities such as another RTB exchange, a mediation platform, or an ad server combines direct campaigns with 3rd party demand in decisioning.

Attribute	Type	Description
fd	Integer; recommended	Entity responsible for the final impression sale decision, where 0 = exchange, 1 = upstream source.
tid	string; recommended	Transaction ID that must be common across all participants in this bid request (e.g., potentially multiple exchanges).
pchain	string; recommended	Payment ID chain string containing embedded syntax described in the TAG Payment ID Protocol v1.0.

ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.3 Object: Regs

This object contains any legal, governmental, or industry regulations that apply to the request. The coppa flag signals whether or not the request falls under the United States Federal Trade Commission's regulations for the United States Children's Online Privacy Protection Act ("COPPA").

Attribute	Type	Description
coppa	integer	Flag indicating if this request is subject to the COPPA regulations established by the USA FTC, where 0 = no, 1 = yes. Refer to Section 7.5 for more information.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.4 Object: Imp

This object describes an ad placement or impression being auctioned. A single bid request can include multiple Imp objects, a use case for which might be an exchange that supports selling all ad positions on a given page. Each Imp object has a required ID so that bids can reference them individually.

The presence of Banner (Section 3.2.6), Video (Section 3.2.7), and/or Native (Section 3.2.9) objects subordinate to the Imp object indicates the type of impression being offered. The publisher can choose one such type which is the typical case or mix them at their discretion. However, any given bid for the impression must conform to one of the offered types.

Attribute	Type	Description
id	string; required	A unique identifier for this impression within the context of the bid request (typically, starts with 1 and increments.
metric	object array	An array of Metric object (Section 3.2.5).
banner	object	A Banner object (Section 3.2.6); required if this impression is offered as a banner ad opportunity.
video	object	A Video object (Section 3.2.7); required if this impression is offered as a video ad opportunity.
audio	object	An Audio object (Section 3.2.8); required if this impression is offered as an audio ad opportunity.
native	object	A Native object (Section 3.2.9); required if this impression is offered as a native ad opportunity.
pmp	object	A Pmp object (Section 3.2.11) containing any private marketplace deals in effect for this impression.
displaymanager	string	Name of ad mediation partner, SDK technology, or player responsible for rendering ad (typically video or mobile). Used by some ad servers to customize ad code by partner. Recommended for video and/or apps.

displaymanagerver	string	Version of ad mediation partner, SDK technology, or player responsible for rendering ad (typically video or mobile). Used by some ad servers to customize ad code by partner. Recommended for video and/or apps.
instl	integer; default 0	1 = the ad is interstitial or full screen, 0 = not interstitial.
tagid	string	Identifier for specific ad placement or ad tag that was used to initiate the auction. This can be useful for debugging of any issues, or for optimization by the buyer.
bidfloor	float; default 0	Minimum bid for this impression expressed in CPM.
bidfloorcur	string; default "USD"	Currency specified using ISO-4217 alpha codes. This may be different from bid currency returned by bidder if this is allowed by the exchange.
clickbrowser	integer	Indicates the type of browser opened upon clicking the creative in an app, where 0 = embedded, 1 = native. Note that the Safari View Controller in iOS 9.x devices is considered a native browser for purposes of this attribute.
secure	integer	Flag to indicate if the impression requires secure HTTPS URL creative assets and markup, where 0 = non-secure, 1 = secure. If omitted, the secure state is unknown, but non-secure HTTP support can be assumed.
iframebuster	string array	Array of exchange-specific names of supported iframe busters.
exp	integer	Advisory as to the number of seconds that may elapse between the auction and the actual impression.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.5 Object: Metric

This object is associated with an impression as an array of metrics. These metrics can offer insight into the impression to assist with decisioning such as average recent viewability, click-through rate, etc. Each metric is identified by its type, reports the value of the metric, and optionally identifies the source or vendor measuring the value.

Attribute	Type	Description
type	string; required	Type of metric being presented using exchange curated string names which should be published to bidders *a priori*.
value	float; required	Number representing the value of the metric. Probabilities must be in the range 0.0 – 1.0.
vendor	string; recommended	Source of the value using exchange curated string names which should be published to bidders *a priori*. If the exchange itself is the source versus a third party, "EXCHANGE" is recommended.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.6 Object: Banner

This object represents the most general type of impression. Although the term "banner" may have very specific meaning in other contexts, here it can be many things including a simple static image, an expandable ad unit, or even in-banner video (refer to the `Video` object in Section 3.2.7 for the more generalized and full featured video ad units). An array of `Banner` objects can also appear within the `Video` to describe optional companion ads defined in the VAST specification.

The presence of a `Banner` as a subordinate of the `Imp` object indicates that this impression is offered as a banner type impression. At the publisher's discretion, that same impression may also be offered as video, audio, and/or native by also including as `Imp` subordinates objects of those types. However, any given bid for the impression must conform to one of the offered types.

Attribute	Type	Description
`format`	object array; recommended	Array of format objects (Section 3.2.10) representing the banner sizes permitted. If none are specified, then use of the `h` and `w` attributes is highly recommended.
`w`	integer	Exact width in device independent pixels (DIPS); recommended if no `format` objects are specified.
`h`	integer	Exact height in device independent pixels (DIPS); recommended if no `format` objects are specified.
`wmax`	integer; DEPRECATED	*NOTE: Deprecated in favor of the `format` array.* Maximum width in device independent pixels (DIPS).
`hmax`	integer; DEPRECATED	*NOTE: Deprecated in favor of the `format` array.* Maximum height in device independent pixels (DIPS).
`wmin`	integer; DEPRECATED	*NOTE: Deprecated in favor of the `format` array.* Minimum width in device independent pixels (DIPS).
`hmin`	integer; DEPRECATED	*NOTE: Deprecated in favor of the `format` array.* Minimum height in device independent pixels (DIPS).
`btype`	integer array	Blocked banner ad types. Refer to List 5.2.
`battr`	integer array	Blocked creative attributes. Refer to List 5.3.
`pos`	integer	Ad position on screen. Refer to List 5.4.
`mimes`	string array	Content MIME types supported. Popular MIME types may include "`application/x-shockwave-flash`", "`image/jpg`", and "`image/gif`".
`topframe`	integer	Indicates if the banner is in the top frame as opposed to an iframe, where 0 = no, 1 = yes.
`expdir`	integer array	Directions in which the banner may expand. Refer to List 5.5.
`api`	integer array	List of supported API frameworks for this impression. Refer to List 5.6. If an API is not explicitly listed, it is assumed not to be supported.

id	string	Unique identifier for this banner object. Recommended when Banner objects are used with a Video object (Section 3.2.7) to represent an array of companion ads. Values usually start at 1 and increase with each object; should be unique within an impression.
vcm	integer	Relevant only for Banner objects used with a Video object (Section 3.2.7) in an array of companion ads. Indicates the companion banner rendering mode relative to the associated video, where 0 = concurrent, 1 = end-card.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.7 Object: Video

This object represents an in-stream video impression. Many of the fields are non-essential for minimally viable transactions, but are included to offer fine control when needed. Video in OpenRTB generally assumes compliance with the VAST standard. As such, the notion of companion ads is supported by optionally including an array of Banner objects (refer to the Banner object in Section 3.2.6) that define these companion ads.

The presence of a Video as a subordinate of the Imp object indicates that this impression is offered as a video type impression. At the publisher's discretion, that same impression may also be offered as banner, audio, and/or native by also including as Imp subordinates objects of those types. However, any given bid for the impression must conform to one of the offered types.

Attribute	Type	Description
mimes	string array; required	Content MIME types supported (e.g., "video/x-ms-wmv", "video/mp4").
minduration	integer; recommended	Minimum video ad duration in seconds.
maxduration	integer; recommended	Maximum video ad duration in seconds.
protocols	integer array; recommended	Array of supported video protocols. Refer to List 5.8. At least one supported protocol must be specified in either the protocol or protocols attribute.
protocol	integer; DEPRECATED	*NOTE: Deprecated in favor of protocols.* Supported video protocol. Refer to List 5.8. At least one supported protocol must be specified in either the protocol or protocols attribute.
w	integer; recommended	Width of the video player in device independent pixels (DIPS).
h	integer; recommended	Height of the video player in device independent pixels (DIPS).
startdelay	integer; recommended	Indicates the start delay in seconds for pre-roll, mid-roll, or post-roll ad placements. Refer to List 5.12 for additional generic values.
placement	integer	Placement type for the impression. Refer to List 5.9.

linearity	integer	Indicates if the impression must be linear, nonlinear, etc. If none specified, assume all are allowed. Refer to List 5.7.
skip	integer	Indicates if the player will allow the video to be skipped, where 0 = no, 1 = yes. If a bidder sends markup/creative that is itself skippable, the Bid object should include the attr array with an element of 16 indicating skippable video. Refer to List 5.3.
skipmin	integer; default 0	Videos of total duration greater than this number of seconds can be skippable; only applicable if the ad is skippable.
skipafter	integer; default 0	Number of seconds a video must play before skipping is enabled; only applicable if the ad is skippable.
sequence	integer	If multiple ad impressions are offered in the same bid request, the sequence number will allow for the coordinated delivery of multiple creatives.
battr	integer array	Blocked creative attributes. Refer to List 5.3.
maxextended	integer	Maximum extended ad duration if extension is allowed. If blank or 0, extension is not allowed. If -1, extension is allowed, and there is no time limit imposed. If greater than 0, then the value represents the number of seconds of extended play supported beyond the maxduration value.
minbitrate	integer	Minimum bit rate in Kbps.
maxbitrate	integer	Maximum bit rate in Kbps.
boxingallowed	integer; default 1	Indicates if letter-boxing of 4:3 content into a 16:9 window is allowed, where 0 = no, 1 = yes.
playbackmethod	integer array	Playback methods that may be in use. If none are specified, any method may be used. Refer to List 5.10. Only one method is typically used in practice. As a result, this array may be converted to an integer in a future version of the specification. It is strongly advised to use only the first element of this array in preparation for this change.
playbackend	integer	The event that causes playback to end. Refer to List 5.11.
delivery	integer array	Supported delivery methods (e.g., streaming, progressive). If none specified, assume all are supported. Refer to List 5.15.
pos	integer	Ad position on screen. Refer to List 5.4.
companionad	object array	Array of Banner objects (Section 3.2.6) if companion ads are available.
api	integer array	List of supported API frameworks for this impression. Refer to List 5.6. If an API is not explicitly listed, it is assumed not to be supported.
companiontype	integer array	Supported VAST companion ad types. Refer to List 5.14. Recommended if companion Banner objects are included via the companionad array. If one of these banners will be rendered as an end-card, this can be specified using the vcm attribute with the particular banner (Section 3.2.6).
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.8 Object: Audio

This object represents an audio type impression. Many of the fields are non-essential for minimally viable transactions, but are included to offer fine control when needed. Audio in OpenRTB generally assumes compliance with the DAAST standard. As such, the notion of companion ads is supported by optionally including an array of `Banner` objects (refer to the `Banner` object in Section 3.2.6) that define these companion ads.

The presence of a `Audio` as a subordinate of the `Imp` object indicates that this impression is offered as an audio type impression. At the publisher's discretion, that same impression may also be offered as banner, video, and/or native by also including as `Imp` subordinates objects of those types. However, any given bid for the impression must conform to one of the offered types.

Attribute	Type	Description
`mimes`	string array; required	Content MIME types supported (e.g., "`audio/mp4`").
`minduration`	integer; recommended	Minimum audio ad duration in seconds.
`maxduration`	integer; recommended	Maximum audio ad duration in seconds.
`protocols`	integer array; recommended	Array of supported audio protocols. Refer to List 5.8.
`startdelay`	integer; recommended	Indicates the start delay in seconds for pre-roll, mid-roll, or post-roll ad placements. Refer to List 5.12.
`sequence`	integer	If multiple ad impressions are offered in the same bid request, the sequence number will allow for the coordinated delivery of multiple creatives.
`battr`	integer array	Blocked creative attributes. Refer to List 5.3.
`maxextended`	integer	Maximum extended ad duration if extension is allowed. If blank or 0, extension is not allowed. If -1, extension is allowed, and there is no time limit imposed. If greater than 0, then the value represents the number of seconds of extended play supported beyond the `maxduration` value.
`minbitrate`	integer	Minimum bit rate in Kbps.
`maxbitrate`	integer	Maximum bit rate in Kbps.
`delivery`	integer array	Supported delivery methods (e.g., streaming, progressive). If none specified, assume all are supported. Refer to List 5.15.
`companionad`	object array	Array of `Banner` objects (Section 3.2.6) if companion ads are available**.**
`api`	integer array	List of supported API frameworks for this impression. Refer to List 5.6. If an API is not explicitly listed, it is assumed not to be supported.
`companiontype`	integer array	Supported DAAST companion ad types. Refer to List 5.14. Recommended if companion `Banner` objects are included via the `companionad` array.
`maxseq`	integer	The maximum number of ads that can be played in an ad pod.

feed	integer	Type of audio feed. Refer to List 5.16.
stitched	integer	Indicates if the ad is stitched with audio content or delivered independently, where 0 = no, 1 = yes.
nvol	integer	Volume normalization mode. Refer to List 5.17.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.9 Object: Native

This object represents a native type impression. Native ad units are intended to blend seamlessly into the surrounding content (e.g., a sponsored Twitter or Facebook post). As such, the response must be well-structured to afford the publisher fine-grained control over rendering.

The Native Subcommittee has developed a companion specification to OpenRTB called the Dynamic Native Ads API. It defines the request parameters and response markup structure of native ad units. This object provides the means of transporting request parameters as an opaque string so that the specific parameters can evolve separately under the auspices of the Dynamic Native Ads API. Similarly, the ad markup served will be structured according to that specification.

The presence of a `Native` as a subordinate of the `Imp` object indicates that this impression is offered as a native type impression. At the publisher's discretion, that same impression may also be offered as banner, video, and/or audio by also including as `Imp` subordinates objects of those types. However, any given bid for the impression must conform to one of the offered types.

Attribute	Type	Description
request	string; required	Request payload complying with the Native Ad Specification.
ver	string; recommended	Version of the Dynamic Native Ads API to which `request` complies; highly recommended for efficient parsing.
api	integer array	List of supported API frameworks for this impression. Refer to List 5.6. If an API is not explicitly listed, it is assumed not to be supported.
battr	integer array	Blocked creative attributes. Refer to List 5.3.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.10 Object: Format

This object represents an allowed size (i.e., height and width combination) or Flex Ad parameters for a banner impression. These are typically used in an array where multiple sizes are permitted. It is recommended that either the `w`/`h` pair or the `wratio`/`hratio`/`wmin` set (i.e., for Flex Ads) be specified.

Attribute	Type	Description
w	integer	Width in device independent pixels (DIPS).
h	integer	Height in device independent pixels (DIPS).
wratio	integer	Relative width when expressing size as a ratio.
hratio	integer	Relative height when expressing size as a ratio.

wmin	integer	The minimum width in device independent pixels (DIPS) at which the ad will be displayed the size is expressed as a ratio.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.11 Object: Pmp

This object is the private marketplace container for direct deals between buyers and sellers that may pertain to this impression. The actual deals are represented as a collection of `Deal` objects. Refer to Section 7.3 for more details.

Attribute	Type	Description
private_auction	integer; default 0	Indicator of auction eligibility to seats named in the Direct Deals object, where 0 = all bids are accepted, 1 = bids are restricted to the deals specified and the terms thereof.
deals	object array	Array of `Deal` (Section 3.2.12) objects that convey the specific deals applicable to this impression.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.12 Object: Deal

This object constitutes a specific deal that was struck *a priori* between a buyer and a seller. Its presence with the `Pmp` collection indicates that this impression is available under the terms of that deal. Refer to Section 7.3 for more details.

Attribute	Type	Description
id	string; required	A unique identifier for the direct deal.
bidfloor	float; default 0	Minimum bid for this impression expressed in CPM.
bidfloorcur	string; default "USD"	Currency specified using ISO-4217 alpha codes. This may be different from bid currency returned by bidder if this is allowed by the exchange.
at	integer	Optional override of the overall auction type of the bid request, where 1 = First Price, 2 = Second Price Plus, 3 = the value passed in `bidfloor` is the agreed upon deal price. Additional auction types can be defined by the exchange.
wseat	string array	Whitelist of buyer seats (e.g., advertisers, agencies) allowed to bid on this deal. IDs of seats and the buyer's customers to which they refer must be coordinated between bidders and the exchange *a priori*. Omission implies no seat restrictions.
wadomain	string array	Array of advertiser domains (e.g., advertiser.com) allowed to bid on this deal. Omission implies no advertiser restrictions.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.13 Object: Site

This object should be included if the ad supported content is a website as opposed to a non-browser application. A bid request must not contain both a `Site` and an `App` object. At a minimum, it is useful to provide a site ID or page URL, but this is not strictly required.

Attribute	Type	Description
`id`	string; recommended	Exchange-specific site ID.
`name`	string	Site name (may be aliased at the publisher's request).
`domain`	string	Domain of the site (e.g., "mysite.foo.com").
`cat`	string array	Array of IAB content categories of the site. Refer to List 5.1.
`sectioncat`	string array	Array of IAB content categories that describe the current section of the site. Refer to List 5.1.
`pagecat`	string array	Array of IAB content categories that describe the current page or view of the site. Refer to List 5.1.
`page`	string	URL of the page where the impression will be shown.
`ref`	string	Referrer URL that caused navigation to the current page.
`search`	string	Search string that caused navigation to the current page.
`mobile`	integer	Indicates if the site has been programmed to optimize layout when viewed on mobile devices, where 0 = no, 1 = yes.
`privacypolicy`	integer	Indicates if the site has a privacy policy, where 0 = no, 1 = yes.
`publisher`	object	Details about the Publisher (Section 3.2.15) of the site.
`content`	object	Details about the Content (Section 3.2.16) within the site.
`keywords`	string	Comma separated list of keywords about the site.
`ext`	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.14 Object: App

This object should be included if the ad supported content is a non-browser application (typically in mobile) as opposed to a website. A bid request must not contain both an `App` and a `Site` object. At a minimum, it is useful to provide an App ID or bundle, but this is not strictly required.

Attribute	Type	Description
`id`	string; recommended	Exchange-specific app ID.
`name`	string	App name (may be aliased at the publisher's request).
`bundle`	string	A platform-specific application identifier intended to be unique to the app and independent of the exchange. On Android, this should be a bundle or package name (e.g., com.foo.mygame). On iOS, it is typically a numeric ID.
`domain`	string	Domain of the app (e.g., "mygame.foo.com").
`storeurl`	string	App store URL for an installed app; for IQG 2.1 compliance.

cat	string array	Array of IAB content categories of the app. Refer to List 5.1.
sectioncat	string array	Array of IAB content categories that describe the current section of the app. Refer to List 5.1.
pagecat	string array	Array of IAB content categories that describe the current page or view of the app. Refer to List 5.1.
ver	string	Application version.
privacypolicy	integer	Indicates if the app has a privacy policy, where 0 = no, 1 = yes.
paid	integer	0 = app is free, 1 = the app is a paid version.
publisher	object	Details about the Publisher (Section 3.2.15) of the app.
content	object	Details about the Content (Section 3.2.16) within the app.
keywords	string	Comma separated list of keywords about the app.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.15 Object: Publisher

This object describes the publisher of the media in which the ad will be displayed. The publisher is typically the seller in an OpenRTB transaction.

Attribute	Type	Description
id	string	Exchange-specific publisher ID.
name	string	Publisher name (may be aliased at the publisher's request).
cat	string array	Array of IAB content categories that describe the publisher. Refer to List 5.1.
domain	string	Highest level domain of the publisher (e.g., "publisher.com").
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.16 Object: Content

This object describes the content in which the impression will appear, which may be syndicated or non-syndicated content. This object may be useful when syndicated content contains impressions and does not necessarily match the publisher's general content. The exchange might or might not have knowledge of the page where the content is running, as a result of the syndication method. For example might be a video impression embedded in an iframe on an unknown web property or device.

Attribute	Type	Description
id	string	ID uniquely identifying the content.
episode	integer	Episode number.
title	string	Content title. *Video Examples:* "Search Committee" (television), "A New Hope" (movie), or "Endgame" (made for web). *Non-Video Example:* "Why an Antarctic Glacier Is Melting So Quickly" (Time magazine article).

series	string	Content series. *Video Examples:* "The Office" (television), "Star Wars" (movie), or "Arby 'N' The Chief" (made for web). *Non-Video Example:* "Ecocentric" (Time Magazine blog).
season	string	Content season (e.g., "Season 3").
artist	string	Artist credited with the content.
genre	string	Genre that best describes the content (e.g., rock, pop, etc).
album	string	Album to which the content belongs; typically for audio.
isrc	string	International Standard Recording Code conforming to ISO-3901.
producer	object	Details about the content `Producer` (Section 3.2.17).
url	string	URL of the content, for buy-side contextualization or review.
cat	string array	Array of IAB content categories that describe the content producer. Refer to List 5.1.
prodq	integer	Production quality. Refer to List 5.13.
videoquality	integer; DEPRECATED	*Note: Deprecated in favor of `prodq`.* Video quality. Refer to List 5.13.
context	integer	Type of content (game, video, text, etc.). Refer to List 5.18.
contentrating	string	Content rating (e.g., MPAA).
userrating	string	User rating of the content (e.g., number of stars, likes, etc.).
qagmediarating	integer	Media rating per IQG guidelines. Refer to List 5.19.
keywords	string	Comma separated list of keywords describing the content.
livestream	integer	0 = not live, 1 = content is live (e.g., stream, live blog).
sourcerelationship	integer	0 = indirect, 1 = direct.
len	integer	Length of content in seconds; appropriate for video or audio.
language	string	Content language using ISO-639-1-alpha-2.
embeddable	integer	Indicator of whether or not the content is embeddable (e.g., an embeddable video player), where 0 = no, 1 = yes.
data	object array	Additional content data. Each `Data` object (Section 3.2.21) represents a different data source.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.17 Object: Producer

This object defines the producer of the content in which the ad will be shown. This is particularly useful when the content is syndicated and may be distributed through different publishers and thus when the producer and publisher are not necessarily the same entity.

Attribute	Type	Description
id	string	Content producer or originator ID. Useful if content is syndicated and may be posted on a site using embed tags.

name	string	Content producer or originator name (e.g., “Warner Bros”).
cat	string array	Array of IAB content categories that describe the content producer. Refer to List 5.1.
domain	string	Highest level domain of the content producer (e.g., “producer.com”).
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.18 Object: Device

This object provides information pertaining to the device through which the user is interacting. Device information includes its hardware, platform, location, and carrier data. The device can refer to a mobile handset, a desktop computer, set top box, or other digital device.

Attribute	Type	Description
ua	string; recommended	Browser user agent string.
geo	object; recommended	Location of the device assumed to be the user’s current location defined by a Geo object (Section 3.2.19).
dnt	integer; recommended	Standard “Do Not Track” flag as set in the header by the browser, where 0 = tracking is unrestricted, 1 = do not track.
lmt	integer; recommended	“Limit Ad Tracking” signal commercially endorsed (e.g., iOS, Android), where 0 = tracking is unrestricted, 1 = tracking must be limited per commercial guidelines.
ip	string; recommended	IPv4 address closest to device.
ipv6	string	IP address closest to device as IPv6.
devicetype	integer	The general type of device. Refer to List 5.21.
make	string	Device make (e.g., “Apple”).
model	string	Device model (e.g., “iPhone”).
os	string	Device operating system (e.g., “iOS”).
osv	string	Device operating system version (e.g., “3.1.2”).
hwv	string	Hardware version of the device (e.g., “5S” for iPhone 5S).
h	integer	Physical height of the screen in pixels.
w	integer	Physical width of the screen in pixels.
ppi	integer	Screen size as pixels per linear inch.
pxratio	float	The ratio of physical pixels to device independent pixels.
js	integer	Support for JavaScript, where 0 = no, 1 = yes.
geofetch	integer	Indicates if the geolocation API will be available to JavaScript code running in the banner, where 0 = no, 1 = yes.
flashver	string	Version of Flash supported by the browser.
language	string	Browser language using ISO-639-1-alpha-2.

carrier	string	Carrier or ISP (e.g., "VERIZON") using exchange curated string names which should be published to bidders *a priori*.
mccmnc	string	Mobile carrier as the concatenated MCC-MNC code (e.g., "310-005" identifies Verizon Wireless CDMA in the USA). Refer to https://en.wikipedia.org/wiki/Mobile_country_code for further examples. Note that the dash between the MCC and MNC parts is required to remove parsing ambiguity.
connectiontype	integer	Network connection type. Refer to List 5.22.
ifa	string	ID sanctioned for advertiser use in the clear (i.e., not hashed).
didsha1	string	Hardware device ID (e.g., IMEI); hashed via SHA1.
didmd5	string	Hardware device ID (e.g., IMEI); hashed via MD5.
dpidsha1	string	Platform device ID (e.g., Android ID); hashed via SHA1.
dpidmd5	string	Platform device ID (e.g., Android ID); hashed via MD5.
macsha1	string	MAC address of the device; hashed via SHA1.
macmd5	string	MAC address of the device; hashed via MD5.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

BEST PRACTICE: There are currently no prominent open source lists for device makes, models, operating systems, or carriers. Exchanges typically use commercial products or other proprietary lists for these attributes. Until suitable open standards are available, exchanges are highly encouraged to publish lists of their device make, model, operating system, and carrier values to bidders.

BEST PRACTICE: Proper device IP detection in mobile is not straightforward. Typically it involves starting at the left of the `x-forwarded-for` header, skipping private carrier networks (e.g., `10.x.x.x` or `192.x.x.x`), and possibly scanning for known carrier IP ranges. Exchanges are urged to research and implement this feature carefully when presenting device IP values to bidders.

3.2.19 Object: Geo

This object encapsulates various methods for specifying a geographic location. When subordinate to a `Device` object, it indicates the location of the device which can also be interpreted as the user's current location. When subordinate to a `User` object, it indicates the location of the user's home base (i.e., not necessarily their current location).

The `lat`/`lon` attributes should only be passed if they conform to the accuracy depicted in the `type` attribute. For example, the centroid of a geographic region such as postal code should not be passed.

Attribute	Type	Description
lat	float	Latitude from -90.0 to +90.0, where negative is south.
lon	float	Longitude from -180.0 to +180.0, where negative is west.
type	integer	Source of location data; recommended when passing `lat`/`lon`. Refer to List 5.20.

accuracy	integer	Estimated location accuracy in meters; recommended when lat/lon are specified and derived from a device's location services (i.e., type = 1). Note that this is the accuracy as reported from the device. Consult OS specific documentation (e.g., Android, iOS) for exact interpretation.
lastfix	integer	Number of seconds since this geolocation fix was established. Note that devices may cache location data across multiple fetches. Ideally, this value should be from the time the actual fix was taken.
ipservice	integer	Service or provider used to determine geolocation from IP address if applicable (i.e., type = 2). Refer to List 5.23.
country	string	Country code using ISO-3166-1-alpha-3.
region	string	Region code using ISO-3166-2; 2-letter state code if USA.
regionfips104	string	Region of a country using FIPS 10-4 notation. While OpenRTB supports this attribute, it has been withdrawn by NIST in 2008.
metro	string	Google metro code; similar to but not exactly Nielsen DMAs. See Appendix A for a link to the codes.
city	string	City using United Nations Code for Trade & Transport Locations. See Appendix A for a link to the codes.
zip	string	Zip or postal code.
utcoffset	integer	Local time as the number +/- of minutes from UTC.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.20 Object: User

This object contains information known or derived about the human user of the device (i.e., the audience for advertising). The user `id` is an exchange artifact and may be subject to rotation or other privacy policies. However, this user ID must be stable long enough to serve reasonably as the basis for frequency capping and retargeting.

Attribute	Type	Description
id	string; recommended	Exchange-specific ID for the user. At least one of `id` or `buyeruid` is recommended.
buyeruid	string; recommended	Buyer-specific ID for the user as mapped by the exchange for the buyer. At least one of `buyeruid` or `id` is recommended.
yob	integer	Year of birth as a 4-digit integer.
gender	string	Gender, where "M" = male, "F" = female, "O" = known to be other (i.e., omitted is unknown).
keywords	string	Comma separated list of keywords, interests, or intent.
customdata	string	Optional feature to pass bidder data that was set in the exchange's cookie. The string must be in base85 cookie safe characters and be in any format. Proper JSON encoding must be used to include "escaped" quotation marks.

geo	object	Location of the user's home base defined by a Geo object (Section 3.2.19). This is not necessarily their current location.
data	object array	Additional user data. Each Data object (Section 3.2.21) represents a different data source.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.21 Object: Data

The data and segment objects together allow additional data about the related object (e.g., user, content) to be specified. This data may be from multiple sources whether from the exchange itself or third parties as specified by the id field. A bid request can mix data objects from multiple providers. The specific data providers in use should be published by the exchange *a priori* to its bidders.

Attribute	Type	Description
id	string	Exchange-specific ID for the data provider.
name	string	Exchange-specific name for the data provider.
segment	object array	Array of Segment (Section 3.2.22) objects that contain the actual data values.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.22 Object: Segment

Segment objects are essentially key-value pairs that convey specific units of data. The parent Data object is a collection of such values from a given data provider. The specific segment names and value options must be published by the exchange *a priori* to its bidders.

Attribute	Type	Description
id	string	ID of the data segment specific to the data provider.
name	string	Name of the data segment specific to the data provider.
value	string	String representation of the data segment value.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.