# Audience Size Forecasting

## Fast and Smart Budget Planning for Media Buyers

Yeming Shi, Claudia Perlich, Rod Hook
Dstillery
New York, NY
yshi,claudia,rod@dstillery.com

Wickus Martin, Melinda Han Williams
Dstillery
wmartin,melinda@dstillery.com

Justin Moynihan, Patrick McCarthy
Dstillery
jmoynihan,pmccarthy@dstillery.com

Peter Lenz, Reka Daniel-Weiner, Roger Cost
Dstillery
plenz,reka,rcost@dstillery.com

## ABSTRACT

A growing proportion of digital advertising slots is purchased through real time bidding auctions, which enables advertisers to impose highly specific criteria on which devices and opportunities to target. Employing sophisticated targeting criteria reliably increases the performance of an ad campaign, however too strict criteria will limit its scale. This raises the need to estimate the number of anticipated ad impressions at a given campaign performance level, thus enabling advertisers to tune the campaign's budget to an optimal performance-scale trade-off. In this paper, we provide a way to estimate campaign impressions given the campaign criteria. There are several challenges to this problem. First, the criteria contain logic to include and exclude combinations of audience segments, making the space of possible criteria exponentially large. Furthermore, it is difficult to validate predictions, because we wish to predict the number of impressions available without budget constraints, a situation we can rarely observe in practice. In our approach, we first treat the audience segment inclusion/exclusion criteria separately as a data compression problem, where we use MinHash "sketches" to estimate audience size. We then model the number of available impressions with a regularized linear regression in log space, using multiplier features motivated by the assumption that some components of the additional campaign criteria are conditionally independent. We construct a validation set by projecting observed RTB data (under real budget constraints) to get impression availability without budget constraints. Using this approach, our average prediction is a factor of 2.2 from the true impression availability, and the deployed product responds to user requests in well under a second, meeting both accuracy and latency requirements for decision making in the execution of advertising campaigns.


*KDD '18, August 19–23, 2018, London, United Kingdom*

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219893

## CCS CONCEPTS

• **Information systems** → **Online advertising**; *Online auctions*; • **Theory of computation** → *Bloom filters and hashing*; Data compression; • **Mathematics of computing** → Density estimation;

## KEYWORDS

Online Advertising; Real-Time Bidding; Bloom Filter; Cardinality Estimation

**ACM Reference Format:**
Yeming Shi, Claudia Perlich, Rod Hook, Wickus Martin, Melinda Han Williams, Justin Moynihan, Patrick McCarthy, and Peter Lenz, Reka Daniel-Weiner, Roger Cost. 2018. Audience Size Forecasting: Fast and Smart Budget Planning for Media Buyers. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219893

## 1 INTRODUCTION

Digital advertising monetizes approximately $200 billion worldwide each year[1]. Historically, brands used to think about targeting in terms audiences — a set of consumers who were considered desirable based on some specific criteria — often in terms of demographics (e.g., women of a certain age range). Such an audience has a certain natural cardinality — typically much larger than the marketing budget for any given advertising campaign. These audiences were assumed to consume certain media, say the "Oprah Winfrey Show", and based on estimated viewer levels, a spot in that show was priced.

Fast forward to programmatic advertising: A growing proportion of digital ad slots are purchased through real time bidding (RTB), where an individual's loading web page or mobile app (the publisher or inventory) announces the availability of an ad slot to an ad exchange. The ad exchange runs an automated auction with multiple potential buyers, who execute ad campaigns on behalf of marketers or advertisers (brands). The buyer with the highest bid typically wins, and the resulting ad showing by the winning buyer is known as an impression.

One of the major reasons for the growing popularity of RTB is the flexibility with which it allows marketers and buyers to decide when, to whom, and in what context their ads should be shown. With the ability to target individuals directly, measurable outcomes (clicks or post-view conversions), and large amounts of data on the individual's digital history, machine learning models

have become the workhorse for predicting the probability of every individual to take the desired action. These models now provide a ranking of the entire population, and in order to maximize performance of a campaign, only those consumers with the highest conversion propensity should be targeted. While conceptually straight forward, it is difficult to connect this understanding with the traditional notion of audiences. Because now a good audience is say the top 1 percentile ranked by a specific model. The optimal size of that percentile has to be a function of the budget the marketer is willing to spend. If it is too big, performance will suffer as lower propensity consumers are entering the audience. If it is too small, the campaign does not deliver and the marketer fails to spend the budget.

To further complicate the issue, campaigns typically have a plethora of additional delivery constraints: marketers can decide to serve only to opportunities appearing in specific geographical locations, only on specific apps, or to certain device types, or any combination of the above. With this flexibility comes an interesting question: under the given campaign criteria, how many impressions is a buyer able to deliver for the (remaining) lifetime of a campaign? Knowing the answer not only greatly helps a marketer plan the budget to spend on a buyer's platform, but also helps a buyer to request a proper budget number or adjust the criteria/audiences if necessary.

We will define **avail** as this campaign capacity under the campaign criteria. Estimating it at low latency and enterprise scale has proven to be a major challenge in the industry. It is of course possible to get an estimate by naive counting (database query) on historical data. However, even on only a 1% subsample, a Hive query takes about twenty minutes and yields poor user experience[1].

Conceptually, avail estimation is similar to a subpopulation density estimation problem. The other approach would therefore be to assume independence among different components in the criteria. While some components can be assumed to be independent, others unfortunately have correlations.

A campaign audience is often a combination of different segments of consumers[2]. The goal of prospecting campaigns is finding new customers for a product or brand, while retargeting is serving ads to past customers. For prospecting, individual devices/cookies are scored by machine learning models into segments that correspond to percentiles. Dstillery builds a large number of models to predict conversion propensities for a large number of desirable outcomes[4, 15]. In fact, we often have multiple models all predicting conversions using different machine learning algorithms, and brands often simultaneously target high-conversion-probability segments given by different models.

Audience segments can also be defined without brand-specific predictive models. Examples include segments of people who have visited a particular store or a particular web site and interest subpopulations, such as technology enthusiasts and health care professionals. As a result, segments can have high membership overlap, which makes the size of the union of several segments different from the sum of individual segment sizes. For example, a high conversion audience segment for hotels can overlap with a generic travelers segment. It is very common for campaigns that the targeted audience is a combinations potentially overlapping segments. We tackle this particular violation of independence by using a *data compression* algorithm which provides *cardinality estimation*.

The methodology we develop for audience size estimation is applicable to a wide range of cardinality estimation problems in a variety of business domains. Our primary use case is the execution of prospecting campaigns. These campaigns can be executed directly by the organization that creates the models and resulting segments. Increasingly, secondary players specialize purely on the execution side and allow the marketer to inject their own audience segments (mostly through CRM integrations) or to buy modeled or otherwise created segments from data providers. This is called audience syndication. Reliable avail estimation becomes even more important when segment creation and execution is decoupled since it is now impossible to directly tune segment sizes to a singular campaign. Audience size estimation not only lets data providers send the desirable campaign scale (number of impressions) information to the platform, but also allows them to create different campaign scales at various campaign performance (number of conversions per impression) levels, so marketers can tune their campaigns to the optimal performance-scale tradeoff.

Another application is the ability to provide marketing insights. Brands are interested in understanding as much as possible about their customers. Such knowledge can be, for example, people who visit a certain store or have this common interest are X times more likely to visit my brand's homepage than an average person. The computation of X involves knowing the size of the intersection of two sets, namely store visitors and homepage visitors.

Our methodology provides a solution to these problems at low response time and enterprise scale.

The rest of the paper is organized as follows. Section 2 captures problem setup and business requirements. Section 3 is an outline of the bidding process. Section 4 formulates the problem and describes challenges and our high-level approach. Section 5 is on feature estimations, Section 6 on model building and evaluation, and Section 7 on live product performance.

## 2 PROBLEM SETUP & BUSINESS REQUIREMENTS

Avail forecasting aims to provide an estimate of an ad campaign's impression delivery capacity on a buyer's platform, given a set of campaign criteria. Campaign criteria are of different nature. The main one is the audience — a selection of consumers to be reached. Other specify constraints under which the ads are delivered. Compaign criteria include

**Audience**

Marketers choose the audience as a combination of consumer segments to target. If the marketer selects many or very large segments, the campaign will reach a very wide audience, but the performance will be low. On the other hand, if the marketer only selects the best segments scored by the campaign's conversion model, the campaign will have limited reach, but a high conversion rate. Avail forecasting is a necessary tool to help a marketer

[1] One might propose to add better technology and hardware to the solution, this however would be notably less cost effective than the solution in this paper.

[2] We use the term segment as some basic set of consumers defined somehow whereas audience is a derived set of consumers that are selected for a specific campaign.

to explore the tradeoff and find a near sweet spot in terms of size and performance even before the start of campaign.

As mentioned, audience segments can have notable membership overlap. Forecasted avails should remain accurate under the selection of overlapping audiences.

Additionally, marketers often not only wish to include segments but also exclude certain subpopulation of consumers. The most common scenario is the exclusion of existing customers of the brand from a prospecting campaign as retargeting campaigns often have different prices and creatives and are executed separately.

It's likely that devices in retargeting segments also score high by the conversion model and therefore will be in the good prospecting segments, resulting in an overlap between inclusion and exclusion segments. Depending on the brand, retargeting segments can be very large and nearly "wipe out" the top 1% prospecting segments, causing severe underdelivery. Avail forecast should remain accurate under those conditions, too.

**Context**

Context refers to the targeting media context such as

(1) environment type, i.e., whether to target opportunities in WEB browsers or in APP or both
(2) device class, for example desktop, smart phone, tablet, or any combination of these
(3) ad type, i.e., whether the ad slot is for a display ad or video ad.

**Geographical area**

The targeting geographical area can be specified in several formats such as country, any combination of states, or any combination of postal codes, etc.

**Viewability**

Viewability is the probability of an impression being rendered in view. Viewability predictions are either obtained through a third party API or modeled from viewability measurement data. In either case, viewability requirement is input by the user as a percentage, namely, I want to target the top X% viewable opportunities. This has an obvious impact on delivery.

**Bid price**

Finally, the last campaign parameter that affects the ability to deliver ads is the bid price since the probability that a bid wins the auction on the ad exchange and becomes an impression is sensitive to bid price. Under otherwise equal circumstances, bidding high will increase delivery. Most execution engines have some levels of bid optimization and will adjust individual bids, but marketers typically set some price for the average or maximum bid they are willing to make. In order to optimize bid prices, Dstillery bundles a campaign's audience segments into smaller groups called spends, with each spend given a custom bid price reflecting the conversion propensity of that spend. We will elaborate on the concept of spend later in the paper.

Given all these components, the avail forecasting product should have an interface where users input the above parameters and can alter them interactively. This suggests that the product has to respond to user requests fast (under 1 second or faster) to make this tool useful. The resulting feedback can either be the total number of deliverable impressions or, given the average bid price, a total budget for the campaign. Our goal is to give users an expected budget estimate that is accurate within a factor of three in most cases. The product also outputs avail as a range centered at our prediction.

## 3 BIDDING PROCESS

In order to provide background for later sections, we provide an overview of the steps that a bid request (BRQ) from an ad exchange passes on the buyer's platform.

When a device goes online, the ad exchange generates a BRQ as an invite for interested buyers to make bids to show their ads in the vacant ad slot. The BRQ is populated with contextual information such as the device id, URL of the publisher, and the geographic location of the device.

Bidder is the buyer's highly sophisticated system that decides what ad to show and bids in real time. The bidder applies a chain of filters on the opportunity, values the opportunity, and responds with a bid. The buyer, executing multiple campaigns on behalf of multiple marketers, conducts a mini-auction within its own system to pick one ad from many ad candidates. The bid winning the internal auction is then sent to the exchange where it competes with other buyers in what we refer to as the external auction.

A high-level view of an industry-general bidder pipeline is shown in Figure 1. A BRQ undergoes the following phases in a typical bidder.

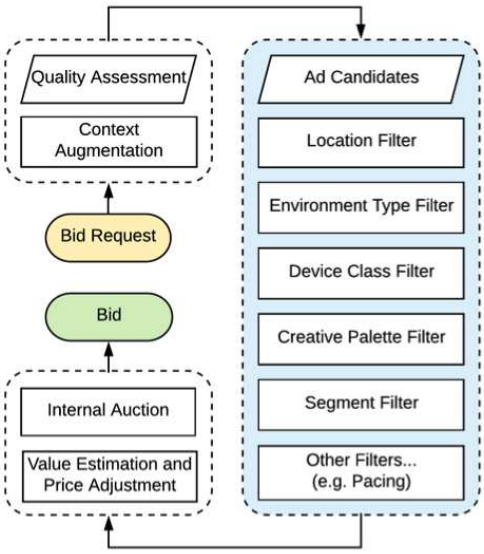


**Figure 1: A high-level view of an industry-general bidder pipeline**

Information Augmentation. The buyer will augment a BRQ's information with additional known facts such as the device's audience segment memberships.

Quality Assessment. Don't bid, for example, if the inventory is of poor quality, for example if the site contains adult content, or if

fraud [18] or header bidding is suspected. In our system, this step eliminates 50% of the billions of BRQs seen daily.

Ad Candidate Selection. The complete list of ad candidates in the system (e.g., spends in Dstillery's system) are considered and evaluated against the opportunity. If an ad candidate's campaign targets, say a specific ad size or postal code, and the BRQ does not meet these requirements, the ad candidate is eliminated from contention.

Value Estimation and Price Adjustment. The forecast of an ad candidate's probability of getting a desired outcome, e.g., a click or conversion, is based on the device's browsing history, the inventory, and how often the device has been shown an ad from the same marketer[3]. Viewability is also evaluated. The selection of viewable opportunities is often implemented as a filter, whereas the click / conversion valuation results in a bid price adjustment[14].

Internal Auction. The highest bid among the remaining ad candidates is sent to the exchange.

Some of the filterings, such as geo filtering, result from the campaign's criteria. Avail forecasting accounts for these filters. Other filters are either modeled as a fractional pass rate or is irrelevant for avail forecasting. For example, pacing filter[12], which rejects opportunities based on the campaign's budget constraint, is irrelevant, since avail forecasting aims at estimating delivery in the absence of budget constraints.

The ad exchange determines which bid wins the external auction, mostly based on bid price.

# 4 PROBLEM FORMULATION, CHALLENGES, AND CONCEPTUAL APPROACH

## 4.1 Problem Formulation

An avail is defined as the impression count projection over a period of time. Under given targeting criteria and bid price, the impression count fluctuation from day to day is small. For simplicity of illustration, we will focus on predicting daily impression delivery in this paper. Each day, we receive a number $N_{BRQ}$ of BRQs[4]. The avail $A$ is simply

$$A(C) = N_{BRQ} \cdot p_i(C) \quad . \tag{1}$$

where $C$ denotes the campaign criteria (audience, context, geographical area, etc.) and bid price, and $p_i(C)$ is the probability that a random BRQ becomes an impression given the criteria and bid price. $N_{BRQ}$ is also a stable number in our system. Therefore, for simplicity of illustration, we will focus on predicting probability $p_i(C)$ from $C$ in the following exposition.

The campaign criteria $C$ are defined as a set of logical operations

$$\begin{aligned} C \ = & AudSeg \ \& \ EnvType \ \& \ DevClass \ \& \ AdType \ \& \\ & Geo \ \& \ Viewability \ \& \ BidPrice \end{aligned} \tag{2}$$

containing *AND's* between audience segments, environment types, device classes, ad type, and geographical areas. Let's elaborate on each.

[3]Often, there are limits (frequency cap) on the number of times a device can be shown ads of the same campaign within a short period of time (one day for example), because the conversion would be attributed to only one impression.

[4]Our system filters poor quality opportunities (adult content sites and fraud) and duplicate opportunities (header bidding). All quantities here, including $N_{BRQ}$, represent numbers after such filtering.

Audience segment is specified as an inclusion segment list with an exclusion segment list. Users target people who fall in any of the segments in the inclusion list but don't belong to any of the segments in the exclusion list. We denote this as a set difference

$$AudSeg \ = \ IncSeg \setminus ExcSeg \quad . \tag{3}$$

Segments *can have overlap*. We write inclusion and exclusion audience segments both as chained lists of logical *OR's*.

$$IncSeg \ = \ IncSeg1 \ \| \ IncSeg2 \ \| \ \ldots \ \| \ IncSegN_i \quad , \tag{4}$$

$$ExcSeg \ = \ ExcSeg1 \ \| \ ExcSeg2 \ \| \ \ldots \ \| \ ExcSegN_e \quad , \tag{5}$$

where $N_i$ and $Ne$ denotes the number of inclusion and exclusion segments, respectively.

Environment type can be either WEB or APP or both. Note that WEB and APP are *disjoint* advertising channels.

Device classes (desktop, smart phone, tablet, etc.) are also *disjoint* and specified as a list.

Ad types (Display vs Video) are *mutually exclusive*. A campaign either serves display ads or video ads but never both.

Geographical area can be specified as follows, 1) an entire country (for example targeting all US) or a list of countries, 2) if one country is selected, the area specification can contain a list of states (in that country), a list of postal codes, or a list of designated market areas (DMAs). In our use case, only one of these three specifications is allowed[5]. Note also that countries, states, postal codes, and DMAs are all *non-overlapping* concepts (for example, no place in a country simultaneously belongs to two states or two postal codes).

The above is the problem that we solve, but our methodology applies to a general class of problems. In the general problem, the "criteria" contain a chain of logical AND operations. Each operand is a chain of OR operations. The operands of some of these OR operations can have set overlap.

## 4.2 Challenges

The problem involves an exponentially large number of possibilities in each dimension or component of the campaign criteria. For example, in the segment dimension, under Equations (4) and (5), there are $2^N - 1$ segment combinations for $N$ segments. As a result, data storage and response time are major challenges. A naive database query approach would take too long. An attempt to store all the counts would be impractical. A feasible approach should featurize the criteria $C$ and compress complex data to reasonable size for storage, and retrieval / estimation should be fast enough to yield sub-second or faster response.

The retrieval algorithm needs to compute engineered features of the criteria $C$, and predict the avail from these features. How to featurize any arbitrary segment combination for an audience, while supporting overlaps and exclusions, is not obvious. For example, when estimating the size of *AudSeg* under the definitions in Equations (3)-(5), if one simply computes the sum of the sizes of $N_i$ inclusion segments and subtracts from that the sum of the sizes of $N_e$ exclusion segments, one would get that, for 40% of the thousands of active spends running on our platform, their audience sizes are negative! A proper estimation should never do that.

[5]Users never ask us to target both states and DMAs in one campaign.

Additional challenges are present in accurately predicting $A(C)$ from the criteria in Equation (2). Some filters in the bidder are not captured by $C$. For example, some campaigns run on selected inventories. Furthermore, it turns out that the probability that a bid wins in the external auction depends on additional factors not captured in $C$, such as the ad exchange.

Finally, it's not clear how to measure $A(C)$ because we don't directly observe the avails of our campaigns and spends but instead observe delivery numbers under campaign budget constraints.

### 4.3 Conceptual Approach

Given the challenges in computing A(C), we make a *conditional independence* assumption. First of all, WEB and APP are *disjoint* advertising environments. If both are selected, our avails calculation is split into two *independent* paths where in each path the avail in one environment is computed, and finally, the result from both paths are added. For simplicity, we will assume only one environment type is selected in the rest of the paper. Our deployed product extends beyond this assumption.

Conditioned on the specified environment type, we assume that segment, (the device class and ad type components of) context, and geo are mutually independent. The reason we condition on environment type is that, different segments can have very different size proportion splits over WEB and APP environments. Some segments are dominantly WEB, some dominantly APP, and others in between. We need to capture the $EnvType$ dependence of $AudSeg$'s size.

Under the criteria of Equation (2), the probability for a BRQ to become an impression is

$$\begin{aligned} p_i(C) \; &= p(EnvType) \cdot p(AudSeg \mid EnvType) \cdot \\ &\quad p(DevClass \,\&\, AdType \mid EnvType) \cdot p(Geo \mid EnvType) \cdot \\ &\quad p(Viewability) \cdot PF(EnvType) \cdot WR_i \cdot WR_e \quad , \end{aligned} \tag{6}$$

where $p(Viewability)$ is simply the top X% viewability targeting percentage that a user specifies, $PF(EnvType)$ represents an average pass factor for filters in the bidder pipeline that are not captured in the criteria $C$ and is modeled just as a function of the environment type[6], and $WR_i$ and $WR_e$ are the probabilities that a bid wins in the internal and external auctions, respectively. Win rates could be functions of features such as bid price and ad type.

Here are a few comments. 1) $EnvType$, $DevClass$, $AdType$ all have low cardinality[7]. In practice, we aggregate all of them in a single context step and call that $p(Context)$. Therefore, Equation (6) becomes

$$\begin{aligned} p_i(C) \; &= p(Context) \cdot p(AudSeg \mid EnvType) \cdot \\ &\quad p(Geo \mid EnvType) \cdot p(Viewability) \cdot \\ &\quad PF(EnvType) \cdot WR_i \cdot WR_e \quad . \end{aligned} \tag{7}$$

2) The conditional independence assumption significantly reduce the amount of data we need to store[8].

[6] $PF(EnvType)$ is different from $p(EnvType)$ which is the probability a BRQ is in the given environment type.

[7] There are less than 50 of all possible ($EnvType$, $DevClass$, $AdType$), i.e., $Context$ combinations.

[8] For example, since there are dozens of contexts and only two environments, $p(Geo \mid EnvType)$ has a much smaller dimension than $p(Geo \mid Context)$.

For $p(Context)$ we just count the fraction of BRQs in the given context. Different device classes have no overlap. When multiple device classes are selected, we just add the probability of each. Because geographical locations have no overlap either, we also just count and add to get $p(Geo \mid EnvType)$.

As mentioned earlier, querying, storing, and looking up all segment union sizes would be impractical. $p(AudSeg \mid EnvType)$ can only be estimated. The problem is to estimate the union size of overlapping sets. For such cardinality estimation problems, various data compression algorithms, for example, Bloom filter [13, 19] and MinHash exist. These algorithms compress individual sets into "sketches" and estimates the union cardinality based on the sketches.

We explicitly model external win rate $WR_e$ because it has high variance. We find the brand's industry sector to be predictive of win rates, so we make industry an input feature.

Combining Equations (1) and (7) we get

$$\begin{aligned} A(C) \; &\propto p(Context) \cdot p(AudSeg \mid EnvType) \cdot \\ &\quad p(Geo \mid EnvType) \cdot p(Viewability) \cdot \\ &\quad WR_i \cdot WR_e \end{aligned} \tag{8}$$

where $N_{BRQ}$ and $PF(EnvType)$ are absorbed into a proportionality constant that only depends on $EnvType$ .

In practice, to estimate $A(C)$ we take the log of both sides of Equation (8) and fit a regularized linear regression model

$$\log A(C) = \sum_k \beta_k x_k + \alpha \tag{9}$$

where every but a few $x_k$ is the log of a term on the RHS of Equation (8), every $\beta_k$ is a coefficient, and $\alpha$ is an intercept that depends only on $EnvType$ . We include categorical industry feature in place of $WR_i$ among $x_k$ . W also include bid price and categorical ad type as features.

Our solution to the challenge of measuring $A(C)$ in the absence of budget constraints is to correct the delivery number for budget constraints. More details are in Section 6.

## 5 ESTIMATIONS

### 5.1 Context and Geo

For context, we aggregate our BRQs by environment type, device class, and ad type. Then $p(Context)$ is just the sum of the proportions of BRQs meeting the context criteria. For geo, we do the same except $p(Geo \mid EnvType)$ is the sum of the *conditional* probabilities of a BRQ to be in the user specified geographical areas.

### 5.2 Segment

Our goal here is to estimate the probability that a BRQ's device belongs to a segment. Segments are defined as collections of browsing devices. If we assume that the number of BRQs per device is insensitive to segment, we can estimate $p(AudSeg \mid EnvType)$ as the probability of a device being in a segment, i.e., # of devices in $AudSeg$ / total # of devices, conditioned on $EnvType$. This significantly reduces the problem size because we see on the order of 100 billion BRQs per day, but on the order of 100 million unique devices per day.

It is impractical to pre-compute the cardinality for all segment combinations. We must estimate. Cardinality estimation typically takes the form of using compressed representations of sets, called sketches[20], in place of entire sets. A sketch retains enough information from a set that one can estimate the size of the set from the sketch. Many types of sketches allow for two sketches to be combined to produce a new sketch that is representative of the union. There are different sketching techniques with different advantages. We've considered two: Bloom filter and MinHash.

A set can be sketched as a Bloom filter, which is comprised of an $m$-sized bit array where the indices of occupied bits are the hashes of set members (entities), in our case, device id's. The bits are initialized to 0 (indicating an empty set), but when an entity is encountered, it is hashed into the range of indices, and the bit at such index is changed to 1, indicating that the entity belongs to the set. Bloom filters are typically used to test for set membership and is applied in distributed database systems.

Bloom filters can also be used to estimate the size of a set. If a (fed) Bloom filter of size (hash range) $m$ that uses one hash functions has $X$ bits set to 1, we can estimate the set cardinality $n$ as [13, 19]

$$n = -m \ln\left[1 - \frac{X}{m}\right] \quad . \tag{10}$$

Note that $X/m$ is the density of occupied bits in the Bloom filter.

Bloom filters can be combined with a bit-wise OR operation to compute the sketch of the union set without losing fidelity. One might expect to be able to combine Bloom filters in a bit-wise AND to compute the Bloom filter for the intersection, but this operation loses fidelity. However, using the set cardinality identity,

$$|A \cap B| = |A| + |B| - |A \cup B| \tag{11}$$

we can estimate the cardinality of the intersection from the union[19]. Similarly, we can use set operations to estimate the cardinality of set difference

$$|A \setminus B| = |A \cup B| - |B| \quad . \tag{12}$$

We use this to estimate the audience size when the campaign criteria contain segment exclusions like in Equation (3).

A Bloom filter becomes "saturated" when too many bits have been set to one, which reduces the accuracy of our cardinality estimation for sets that are large relative to size of the Bloom filter. This is problematic because our segments may be very large, sometimes close to 100 million members, which would require a Bloom filter of more than 100 million bits, or around 12.5 MB in disk space. This adds up quickly, to 200 GB for all the segments in our system. We decide against using Bloom filters for our sketches because MinHash sketch offers advantages.

MinHash is another sketching approach. Feeding a MinHash sketch is similar to feeding a Bloom filter. For each entity we hash it into a hash range, but we only keep the $p$-smallest hash values in a reservoir. If the set size is less than $p$, we keep all hashes. The MinHash sketch is comprised entirely of this reservoir of hash values.

We can take advantage of keeping only the least hash values for our reservoir to estimate cardinality from a MinHash sketch. If a MinHash reservoir contains $q$ ($q \leq p$) integers, and the greatest of them is $M$, the "density" of occupied bits can be estimated as[9] $(q-1)/M$. Using this estimate and Equation (10), we can estimate the set size as[10]

$$n = -m \ln\left[1 - \frac{q-1}{M}\right] \quad . \tag{13}$$

Because for MinHash we store the smallest $p$ hashes rather than the entire $m$-size bitarray, we can set $m$ really large. This dilutes the density of occupied bits, reduces hash collisions, and therefore increases estimation accuracy. A MinHash sketch "saturates" much more slowly than a Bloom filter, because of the much greater hash range that can be used, so MinHash sketches can be used to estimate the size of a much larger set than Bloom filters under equal storage cost. In our system, we used MinHash sketches of reservoir size $p$=100, and all sketches took just 40 MB to store, as opposed to the Bloom filter sketches which took 200 GB.

In order to test the accuracy of Equation (13), for each (single) segment on WEB, we counted its true device count, estimated the count with MinHash, and compared the two. Figure 2 plots, on a log scale, the MinHash estimated segment size against the true segment size (blue). The plot contains 75913 points. A hypothetical perfect estimation which always estimates the exact truth is shown (red) for comparison. It's clear that Equation (13) gives very good segment size estimations.

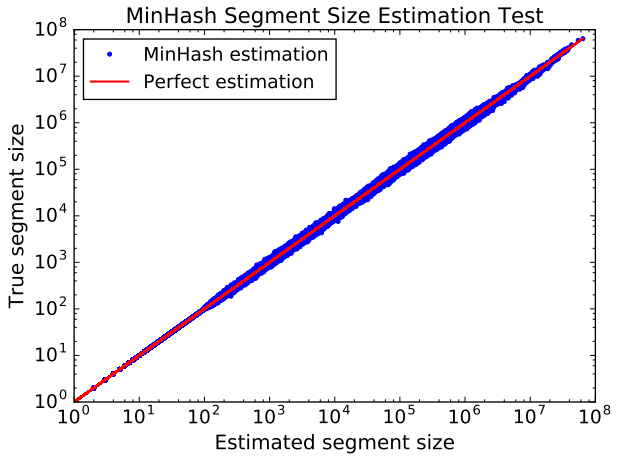


**Figure 2: Testing accuracy of MinHash cardinality estimation on single segments. Truth vs. MinHash estimation is shown in blue. An estimation that always equals to truth is shown in red.**

MinHash sketches for sets can be combined to generate the sketch of the union, by simply "set-union"-ing the reservoirs and keeping the (up to) $p$ smallest hashes. MinHash and Bloom filter both have distributive property over the union operation. The "union" of the MinHashes of individual sets is equal to the MinHash of the union of individual sets. Therefore, MinHashes don't lose fidelity under union operations. Experiments over simulated data shows 3% cardinality estimation error of the union of 100 sets

[9] We assume the integer hash value starts at zero.
[10] If $q < p$, we don't need to estimate, and we know $n = q$ .

whose sizes are distributed similarly to segment sizes in our system. Similar to Bloom filters, intersecting Minhash sketches hurts fidelity, so Equation (11) is used to estimate the size of set intersections. We use Equation (12) to estimate audience size in the presence of segment exclusions.

### 5.3 Internal Win Rate

Internal win rate is the probability that the ad candidate gets selected over competitor ad candidates from a different campaign or different spend. We collect spend level aggregated data before and after the internal auction and find that the average internal win rate on our platform is 32%, much higher than the average external win rate of 4%. Furthermore, we find a trend that, over time, certain industries, such as retail and healthcare (travel and non-profit) often have higher (lower) internal win rates. Therefore, industry feature is used in the final model to capture this trend.

### 5.4 External Win Rate

The most important factor in winning the external auction is bid price. A gradient boosting[7, 8] model trained to predict external win rate revealed a few other quite important features, such as the ad exchange. Some of these, such as ad exchange, are not inputs that user specify. The complex interactions between the ad exchange, publisher, and the marketer of the bid [17] presents another challenge to accurately predicting external win rate.

Our approach goes as follows. The range of past bid prices (per thousand transactions) offered up to the exchange are broken into 10 cent buckets. For each bucket, the number of bid counts and impression counts over the past week are aggregated. Their ratio is calculated as the external win rate. The goal here is to find a function that fits the measurements. The measured data points show, as expected, a monotonic relationship between bid price and win rate. However, the relationship was clearly nonlinear and so the task becomes finding a curve that adequately fits the measurements. Video opportunities are much more expensive to win than display opportunities, ant therefore deserve separate treatment. Video and display win rate curves differ in that display win rates are linear at low bid prices, whereas video win rates stay close to zero and don't show significant increase until bid price is above some threshold. Both curves slowly saturates as the bid price increases. We find that no function fits both Video and Display, so each requires a distinct fit.

We fit the display win rate curve with Equation (14) where $p$ is bid price, and $\gamma$ and $c$ are fitting parameters. This function has the desirable property that it is linear at low $p$ and saturates at $\gamma$ as $p \to \infty$ and is introduced in [23].

$$WR_e(p) = \frac{\gamma \cdot p}{c + p} \tag{14}$$

We fit the video win rate curve with Equation (15) with parameters $\gamma$, $c$, and $\theta$. This function has the desirable property that it is exponentially small at low $p$ and saturates at $\gamma$ as $p \to \infty$.

$$WR_e(p) = \frac{\gamma}{1 + \exp\left[-\theta \cdot (p - c)\right]} \tag{15}$$

The optimal fitting parameters $\gamma$ and $c$ take different values for Display and Video.

Figure 3 shows the external win rate fit as functions of bid price for Display and Video[11].

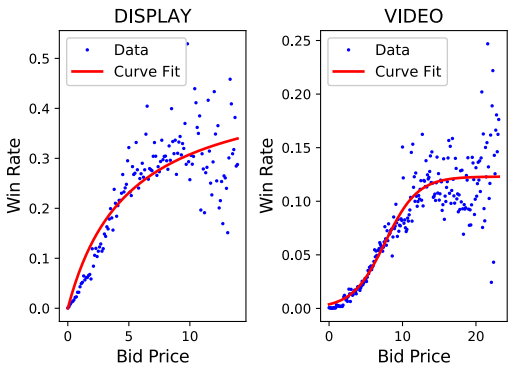


**Figure 3: External win rate fits as functions of bid price per thousand transactions.**

Seasonal factors affect win rate. For instance, near Black Friday and Christmas, the market place gets more competitive and win rates typically drop for the same price level. Both functions are refitted on a weekly basis to allow for changes in market behavior. The fitted parameters change slightly over time.

## 6 MODEL BUILDING AND EVALUATION

Collecting avail data for all possible combinations of campaign criteria would be impossible. Constructing a large number of test campaigns, running them as experiments, and measuring their avails would be too costly for our business. Therefore, we evaluate our model based on the campaigns already running on our platform. Spend is the central entity for the evaluation of avail forecasting. A spend is any number of audience segments grouped together for targeting and is basically a mini-campaign. A spend can also have geo targeting and device-type criteria on it. It has a set bid price, which reflects the relative performance of the spend. A campaign has multiple spends, for example, retargeting spends and prospecting spends, etc. As mentioned earlier, even for existing spends there's challenge in measuring their avails.

### 6.1 Data Collection and Feature Engineering

Preparing data for model building and evaluating model performance is nontrivial because under campaign budget constraints, it's not clear what the true avail is that we should predict. Clearly, we should not fit our model to the number of impressions we observe in each spend. In our system, every campaign's budget is allocated over the duration of the campaign into daily budgets. The pacing filter controls how many opportunities each campaign gets based on its budget constraint, and these opportunities are open to the best performing spends in the campaign, with the rest of the

[11] In each plot, some data points, especially at high prices, are far from the fitted curve. External win rates are affected by factors beyond bid price, such as ad exchange and campaign. Exchange is not an input of our problem, because campaigns run on multiple exchanges. Outliers could be due to a single campaign running at a specific bid price. Our inclusion of industry feature captures campaign specific effects to some degree. Also, we bid predominantly at low price.

spends partially or completely throttled. Everyday for each spend, we aggregate and log pacing filter's throttle attempts $N_{attempts}$ (# of opportunities reaching the pacing filter) and passes (# of opportunities the pacing filter lets through). We can use them to calculate a throttle pass rate

$$tpr = N_{passes}/N_{attempts} \quad , \tag{16}$$

which is a daily aggregate number and is close to 1 for well performing spends, between 0 and 1 for medium performing spends, and close to 0 for underperforming ones. We use $tpr$ to correct the spend's daily impression count $N_{imp}$ to get a spend's impression avail in the absence of budget constraint, i.e.,

$$A(C) = N_{imp}/tpr \quad . \tag{17}$$

This $A(C)$ is the quantity that we'll predict with our model Equation (9). It is worth noting that many spends in our system have very few $N_{passes}$ or very few $N_{imp}$. These small numbers appearing in Equations (16) and (17) increase the variance or error[12] of the dependent variable $A(C)$. For this reason, we discard spends with $N_{passes}$ or $N_{imp}$ less than 10 in our data.

Every day, we collect and stage yesterday's BRQ counts aggregated by context / geo, yesterday's segment MinHashes, win rate over the past week, and yesteday's $N_{imp}$ and $tpr$ . We generate the dependent variable $A(C)$ and model features $x_k$ , including MinHash segment size estimates and fitted external win rates, based on these staged data. A ridge linear regression model of Equation (9) is then fit.

## 6.2 Model Fitting and Results

To prevent overfitting, data is split into training and test sets. The L2-regularization strength of the linear model is tuned using cross validation solely based on training set data.

Below are model fitting results on a typical day. There are 558 active spends on our platform that meet the minimal impression count requirement. The training set contains 390 (70%) spends, and the test set contains 168 (30%) spends. The linear model of Equation (9) gets an $R^2$ metric of 0.80 on the training set and an $R^2$ metric of 0.73 on the test set.

Table 1 shows coefficients $\beta_k$ of continuous features in the model of Equation (9). All continuous features have positive coefficients as expected[13]. In particular, several of them have coefficients close to 1 as would be the case under Equation (8). The categorical industry and ad type features get a range of coefficients from positive to negative. Adding $\log(BidPrice)$ as a feature doesn't improve $R^2$, so we leave $\log(BidPrice)$ out from the model.

**Table 1: Coefficients of continuous features in the model**

| Feature | Coefficient |
|---|---|
| $\log p(Context)$ | 2.81 |
| $\log p(Viewability)$ | 1.28 |
| $\log WR_e$ | 1.05 |
| $\log p(AudSeg \mid EnvType)$ | 0.68 |
| $\log p(Geo \mid EnvType)$ | 0.58 |

[12]Both $N_{passes}$ and $N_{imp}$ are counts, and therefore their variances have a square root relation with their values under Poisson statistics. Their relative variances could be large when their values are small. These relative errors propagate to $A(C)$.
[13]All continuous features have Z-scores greater than 4 except $\log p(Context)$.

Figure 4 is a plot of the log of the true avails (in base 10), $\log A(C)$, against model predicted log of avails for all active spends in our system. Blue (red) dots represent training (test) set samples. There's a clear positive correlation between model prediction and truth. The average the difference between predicted and true log of avails is 0.35, which means on average our prediction is a factor of $10^{0.35} = 2.2$ from the true avails. This is good enough from a product perspective as we aim to give ballpark avail estimates.

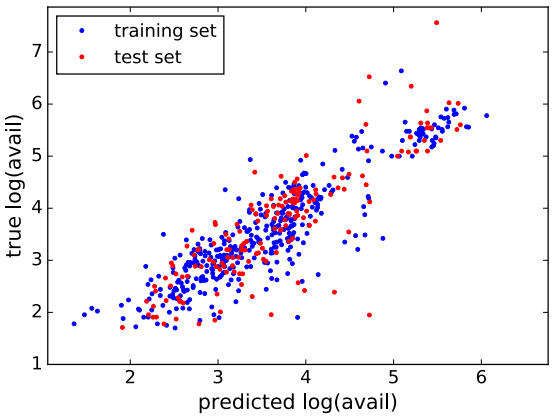


**Figure 4: A plot of true avails versus predicted avails. Both are in base 10 log scale. Blue (red) dots represent training (test) set samples.**

For comparison, the test-set $R^2$ using a linear model with simple features, namely the same features except with 1) each spend's $p(AudSeg \mid EnvType)$ estimated as the sum of its inclusion segment sizes rather than estimated from MinHash and 2) $\log(BidPrice)$ to represent the win rate instead of the curve-fitted $\log WR_e$ , is also 0.73 . We suspect that, although MinHash should give a better $p(AudSeg \mid EnvType)$ estimation, the noise in $A(C)$, due to other bidder pipeline filters not captured by the inputs in Equation (2) and factors beyond the inputs that affect $WR_e$, may obscure the benefit of using MinHash.

# 7 LIVE PERFORMANCE EVALUATION

The avail forecasting tool is deployed as a product for our clients. The product's pipeline starts with ETL in Spark SQL and HiveQL, which process our logged BRQ and bid data, impression data, and device segment data (all in Hive tables). The curve fitting and machine learning are done with SciPy and Scikit-learn. Features and model coefficients are saved to database. A Java micro-service

loads them into memory[14], computes probability multipliers, including $p(AudSeg \mid EnvType)$ which is estimated with MinHash, and then computes the avail result.

### 7.1 User Feedback

Our product, account management, and sales teams have given much positive feedback. Our colleagues used the product in pre-sales processes to confirm how much budget we feel comfortable spending in certain geographical locations and on various segments. Our external clients asked for estimates and we were able to deliver the estimated numbers. The product computes avail numbers that are accurate (after observed impression counts are converted to avail counts) and realistic. The product has been a huge time saver. Until the product went live, our colleagues had to pull numerous charts, calculate many estimations, and take a leap of faith to create an avail estimate. Now it's fast and as simple as a few clicks!

In more recent weeks we hear about increased revenue being tied to this product. A major revenue driver for our company is the amount of incremental budget we can take on mid-way through a campaign. Based on what audiences are performing well, our account managers can use the tool to justify that the client can and should spend more budget in order to increase their returns on investment.

### 7.2 Runtime Comparison

Our existing implementation is fast, returning responses in less than 0.1 second. This is largely due to the two simplifying treatments, conditional independence and MinHash cardinality estimation.

Conditional independence makes the underlying data set size far smaller than the actual Cartesian product of the dimensions. Without the independence simplification we have to traverse every row of BRQ data, check it against user's campaign criteria to determine if it should be included in the estimate, and then sum it into the total. With conditional independence approach, a typical aggregated data set has 60,000 rows of geo data and 50 rows of context data. Without it we would have to traverse $60,000 \times 50 = 3,000,000$ rows of data even just for these two dimensions.

Estimating cardinality via MinHash is also beneficial to the response time. Firstly, retrieving members of segments containing millions of devices from database involves reading a large file. In contrast, MinHash compresses the device list into 100 integers. Secondly, computing the size of segment unions is much faster with MinHash. In experiments on randomly simulated segments averaging 250,000 devices, computing the union of 10 segments using set operations averaged 791 milliseconds, versus 445 microseconds for MinHash estimates.

Another impactful assumption is that segment membership can be estimated independently of geo or context. If we were to instead consider a Cartesian product of segment, geo, and context, with 75,000 segments in our system, we would be left with an aggregation over essentially the full 100 billion BRQs in our system.

[14] Segment MinHashes are loaded to a binary on-disk-hash(ODH) file which the micro-service looks up in real time. We could easily host MinHashes in memory if we want to — the only reason we went with ODH file is that in an earlier design we considered Bloom filters which would be too big to keep in memory.

There are products in the market that can run aggregations over data at this scale in a short time. Google's BigQuery, can aggregate over an 85-billion-row dataset in between 10 and 100 seconds [16]. A typical production setup of Apache Druid, querying a 50-billion-row dataset, returns 90% of queries in less than 1 second [22]. Another option is to use Apache Parquet[5] for storage and Apache Arrow[6] for memory.

While these query times may fall in an acceptable range for users, the operating costs are far beyond those of our solution. BigQuery is a paid product, and Druid, while free, was set up in the example case with sufficient Amazon EC2 resources to run 1302 concurrent threads. By contrast, our production deployment consists of two instances (to avoid machine failure) of lightweight Java micro-service, accessing all of its data either within its 4GB heap memory or through retrievals from local disk, and running on 8-core machines. Apache Parquet, on the other hand, requires pre-processing overhead, namely building the data cube and the associated time cost.

## 8 RELATED WORK

There isn't much work on this problem in literature. The most comparable work we find is [21] by GumGum. They too are interested in the number of advertising opportunities available to their clients. The difference is that they are interested in the avail breakdown mostly by inventory, whereas we aggregate avail in more granular dimensions. While we count opportunities and devices by several dimensions and project them via multipliers into estimated impressions, they count impressions by inventory. They too use MinHash for count estimation. They maintain the device ids from which the hashes come, join them back to features on devices and project on those features before performing their forecast, whereas we maintain distinct MinHashes for each segment.

[3] solves a problem that is similar to part of our problem, namely the segment size estimation. The authors there are interested in estimating the number of documents matching a query expressed in terms of boolean operations on substrings. They didn't adopt the Bloom filter method because it requires large storage space. Instead, they used another technique called min-wise independent permutations which needs a collection of hash functions. Also, since their problem deals with search strings, they use a trie data structure.

We also find that segment-specific scale estimation has application in the cable TV industry. Modern technology allows for direct data collection from TV subscriber devices and the creation of more granular segments with size estimates [2]. Audience size estimates allows TV advertising inventory to be divided into subsets, priced appropriately in advance, and sold to multiple advertisers for the same show and time slot.

Our problem is different from demand forecasting where the change of demand over time is more interesting. As an example, ride share services gain from predicting demand. In a post to Uber's Engineering Blog[11], the authors describe a method to model total demand for Uber rides over the course of a calendar year. They developed an approach in which Long Short Term Memory (LTSM)[9, 10] neural networks were trained on sliding windows of training data. The ability of the LTSMs to relate information across

non-consecutive time points as well as to "forget" gave their model flexibility to predict normal demand as well as demand around unusual events such as Christmas Day.

## 9 CONCLUSIONS AND FUTURE WORK

By dividing the problem into independent steps and using MinHash algorithm and machine learning, we can provide ballpark accurate avail estimates. The response time of the deployed product is fast. We receive positive feedback from users, citing that the product provides accurate forecasts and is a huge time saver for campaign budget planning. The product has led to increased revenue.

In the future, we would like to investigate ways to improve model $R^2$. One idea is to consider adding other parts of the campaign criteria, such as the campaign's frequency cap which may impact impression count, to model inputs.

A separate direction is to make the audience segment size estimation support other logical combinations of segments, namely intersections. This would be useful for marketers who want to target devices in multiple segments simultaneously. We know that MinHash gives a good estimate on the intersection of two sets as long as the set overlap is not too small. When the overlap is small, assuming independence and estimating the joint probability as a product of probability multipliers could give a better result.

It is noteworthy that the audience size estimation methodology in this paper is useful for several new products that are under active development at Dstillery. One example is the "audience studio", where advertisers create their own targeting audiences from any combination of our existing segments, the product provides estimations of the sizes of created audiences, and the created audiences are then available on our partner buyer's platform for ad campaign execution.

## ACKNOWLEDGMENTS

The authors would like to thank Prof. Foster Provost from NYU for insightful discussions. We would also like to thank product managers Mark Jung and Joshua Stewart for collecting user feedback from our account management and sales teams.

## REFERENCES

[1] [n. d.]. Digital advertising spending worldwide from 2015 to 2020 (in billion U.S. dollars). ([n. d.]). Retrieved Feb 10, 2018 from goo.gl/fN9e1T

[2] 2016. Addressable TV Advertising: Creating a Better, More Personal TV and Video Experience. (2016). Retrieved Feb 10, 2018 from goo.gl/aJLj11

[3] Zhiyuan Chen, Nick Koudas, Flip Korn, and S. Muthukrishnan. 2000. Selectively Estimation for Boolean Queries. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '00)*. ACM, New York, NY, USA, 216–225. https://doi.org/10.1145/335168.335225

[4] Brian Dalessandro, Daizhuo Chen, Troy Raeder, Claudia Perlich, Melinda Han Williams, and Foster Provost. 2014. Scalable hands-free transfer learning for online advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 1573–1582.

[5] Apache Software Foundation. 2014. (2014). Retrieved May 17, 2018 from https://parquet.apache.org

[6] Apache Software Foundation. 2017. (2017). Retrieved May 17, 2018 from https://arrow.apache.org

[7] Jerome H. Friedman. 2001. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics* 29, 5 (October 2001), 1189–1232.

[8] Jerome H. Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38, 4 (February 2002), 367–378.

[9] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to Forget: Continual Prediction with LSTM. *Neural Computation* 12 (1999), 2451–2471.

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (1997), 1735–1780.

[11] Nikolay Laptev, Slawek Smyl, and Santhosh Shanmugam. 2017. Engineering Extreme Event Forecasting at Uber with Recurrent Neural Networks. (2017). Retrieved Feb 09, 2018 from https://eng.uber.com/neural-networks/

[12] Kuang-Chih Lee, Ali Jalali, and Ali Dasdan. 2013. Real time bid optimization with smooth budget delivery in online advertising. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*. ACM, New York, NY, USA.

[13] Odysseas Papapetrou, Wolf Siberski, and Wolfgang Nejdl. 2010. Cardinality estimation and dynamic length adaptation for Bloom filters. *Distributed and Parallel Databases* 28, 2-3 (December 2010), 119–156.

[14] Claudia Perlich, Brian Dalessandro, Rod Hook, Ori Stitelman, Troy Raeder, and Foster Provost. 2012. Bid optimizing and inventory scoring in targeted online advertising. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 804–812.

[15] Troy Raeder, Ori Stitelman, Brian Dalessandro, Claudia Perlich, and Foster Provost. 2012. Design principles of massive, robust prediction systems. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 1357–1365.

[16] Kazunori Sato. 2012. An Inside Look at Google BigQuery. (2012). Retrieved Jan 29, 2018 from https://cloud.google.com/files/BigQueryTechnicalWP.pdf

[17] Yeming Shi, Ori Stitelman, and Claudia Perlich. 2017. Blacklisting the Blacklist in Online Advertising: Improving Delivery by Bidding for What You Can Win. In *Proceedings of the ADKDD'17*. ACM, New York, NY, USA, 1:1–1:6.

[18] Ori Stitelman, Claudia Perlich, Brian Dalessandro, Rod Hook, Troy Raeder, and Foster Provost. 2013. Using co-visitation networks for classifying non-Intentional traffic. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 1240–1248.

[19] S. Joshua Swamidass and Pierre Baldi. 2007. Mathematical correction for fingerprint similarity measures to improve chemical retrieval. *Journal of Chemical Information and Modeling* 47, 3 (April 2007), 952–964.

[20] Daniel Ting. 2016. Towards optimal cardinality estimation of unions and intersections with sketches. In *Proceedings of the 22th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 1195–1204.

[21] Michael Williams. 2016. An Ad Impression Forecasting Tool Built with Apache Spark. (2016). Retrieved Feb 10, 2018 from goo.gl/mYbYuJ

[22] Fangjin Yang, Eric Tschetter, Xavier Léauté, Nelson Ray, Gian Merlino, and Deep Ganguli. 2014. Druid: A Real-time Analytical Data Store. In *Proceedings of the 2014 ACM SIGMOD international conference on management of data*. ACM, 157–168.

[23] Weinan Zhang, Shuai Yuan, and Jun Wang. 2014. Optimal real-time bidding for display advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 1077–1086.