Vungle Inc. Improves Monetization Using Big Data Analytics

Bert De Reyck
UCL School of Management, University College London, London, United Kingdom, bdereyck@ucl.ac.uk
Ioannis Fragkos
Department of Technology and Operations Management, Rotterdam School of Management, Rotterdam, The Netherlands, fragkos@rsm.nl
Yael Grushka-Cockayne, Casey Lichtendahl
Darden School of Business, University of Virginia, Charlottesville, Virginia 22903, {grushkay@darden.virginia.edu, lichtendahlc@darden.virginia.edu}
Hammond Guerin
Data Science Team, Vungle Inc., San Francisco, California 94107, hammond.guerin@vungle.com
Andrew Kritzer
BookMD, Los Angeles, California 90081, akritzer@gmail.com

The advent of big data has created opportunities for firms to customize their products and services to unprecedented levels of granularity. Using big data to personalize an offering in real time, however, remains a major challenge. In the mobile advertising industry, once a customer enters the network, an ad-serving decision must be made in a matter of milliseconds. In this work, we describe the design and implementation of an ad-serving algorithm that incorporates machine-learning methods to make personalized ad-serving decisions within milliseconds. We developed this algorithm for Vungle Inc., one of the largest global mobile ad networks. Our approach also addresses other important issues that most ad networks face, such as user fatigue, budget restrictions, and campaign pacing. In an A/B test versus the company's legacy algorithm, our algorithm generated a 23 percent increase in revenue per 1,000 impressions. Across the company's network, this increase represents a $1 million increase in monthly revenue.

Keywords*:* mobile advertising • logistic regression • big data • feature selection • computational advertising • machine learning • cloud computing.

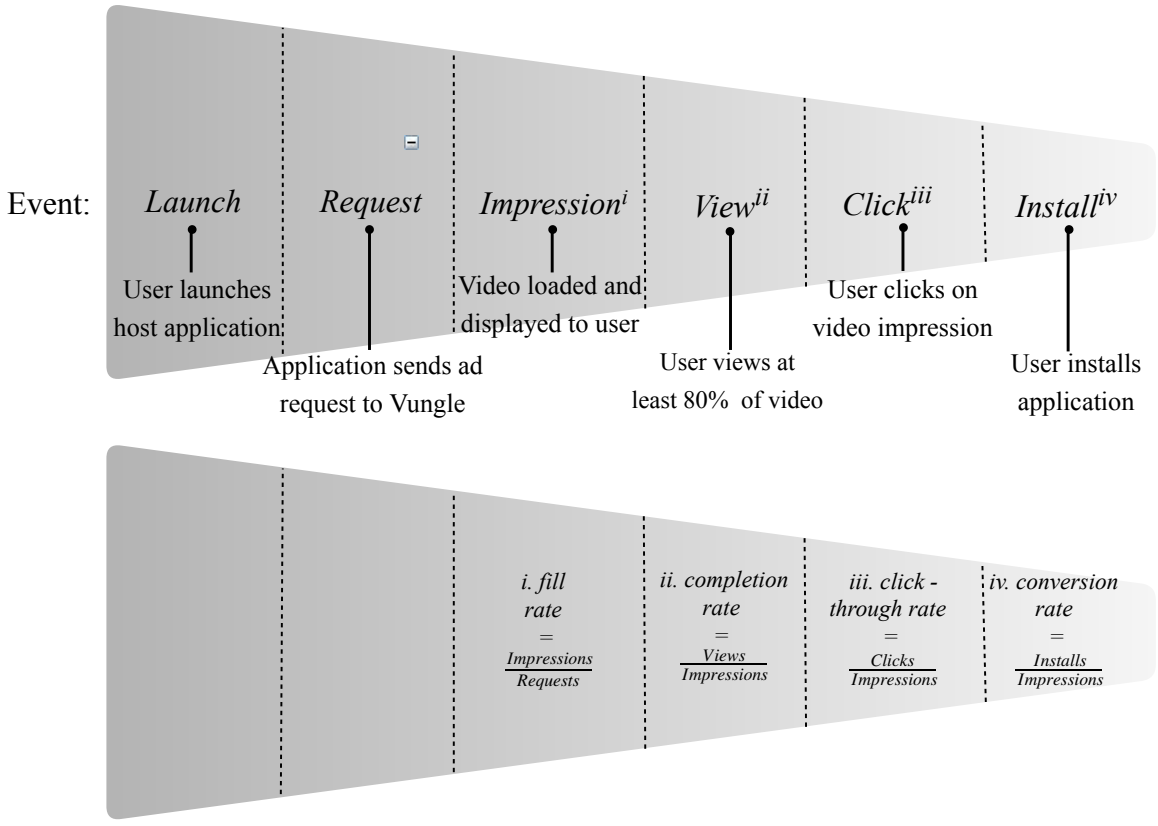
Mobile advertising is a flourishing business. With more than 230 billion application downloads from the leading app stores in 2014, and more than a trillion downloads expected by 2020 (Gandhi 2015), mobile devices have revolutionized the landscape of the advertising industry. The vast amount of data available to advertisers and advertising networks creates opportunities for them to target customers with a granularity that was impossible before the information age (Balseiro et al. 2015). However, utilizing the massive amount of available data to implement efficient monetization policies is a nontrivial problem. A focal decision problem of mobile advertising networks is *ad serving* (i.e., determining which ad to serve to each of the thousands of ad requests that arrive each second from mobile devices worldwide). To ensure a smooth user experience, ad-serving decisions need to be made in less than 50 milliseconds (ms), six times less than it takes the human eye to blink.

This is a challenge for Vungle Inc., one of the largest global mobile advertising networks, which displays more than 1.5 billion video ads to more than 200 million people per month (Guerin 2016). Vungle was founded in 2011 by two young entrepreneurs from the United Kingdom, originally as a video ad production firm. The turning point for Vungle came in 2012, when its two founders creatively used their own video production technology to gain the attention of AnglePad, the San Francisco-based start-up incubator. In doing so, they surpassed 2,000 other applicants to win the final slot in an incubator program. Today, having raised more than $25 million in capital, Vungle is frequently listed among the most high-potential start-ups in Silicon Valley (Ha 2014). Currently, it employs more than 160 people, many of whom are software engineers, business developers, and data scientists, located in six countries across the globe.

The core service that Vungle offers is a platform that enables user acquisition (UA); that is, advertisers embed their video ads within popular mobile gaming applications (i.e., publisher

apps) to encourage users to download and install new apps. Advertisement slots are generated as users interact with the publisher apps, either after predefined events or after user-triggered events. For example, an ad slot may be generated when a user has completed a stage in the publisher app, and before the new stage is loaded. In other cases, publisher apps allow users to gain some amount of virtual currency if they decide to watch a video ad. Regardless of the specific ad generation mechanism, Vungle allocates a video ad from a list of eligible candidates for the generated slot. Figure 1 shows an overview of the main events that occur during the advertising operation and reports the most important performance metrics.

Figure 1. In This Graphic Of Events And Performance Metrics In Mobile Advertising, The Funnel Illustrates The Reduction In The Number Of Generated Events Over Time



Note. The fill rate depends on the network's ability to fulfill the requested ad slots, while the completion, click-through, and conversion rates depend on user actions. Pricing models can be based

on pay-per-impression (CPM), pay-per-completed-view (CPCV), pay-per-click (CPC), or pay-per-install (CPI), the latter being most prominent.

Ads can be monetized across different stages of the advertising funnel: when (1) a video impression is loaded and displayed to a user, (2) a user watches at least 80 percent of a video, (3) a user clicks on the video impression and is redirected to the app store, or (4) a user installs the advertised application. The corresponding performance metrics, that is, the fill, completion, click-through, and conversion rates, respectively, are linked directly to advertising revenues, measured per 1,000 impressions and denoted as expected revenue per mille (eRPM).

In this paper, we describe the conceptualization, development, and deployment of an ad-serving algorithm, which we developed for Vungle and which resulted in an estimated increase in monthly revenues of around $1 million. The backbone of our system is a family of regularized logistic regression models, which predict, for each candidate ad, the probability of a revenue-generating user action. The system then calculates the expected profit from displaying each ad and makes an allocation decision, all in less than 50 ms to enable an instantaneous response. Because our implementation considers subtle user features, such as user fatigue, installation history, and that user's interaction history with publisher applications, it significantly improves performance. Our approach includes the following innovative features.

- It incorporates a user-specific, real-time procedure that strikes a balance between sending a large variety of ads and sending ads of high quality, considering the user's level of engagement with the host application. This diversification mechanism exposes users who are highly engaged with the host application to a larger variety of ads, by taking into account their recent interactions with ads and signs of fatigue. For a user with lower engagement,

our procedure rotates ads at a lower speed, and gives higher selection probabilities to higher-quality ads.

- It addresses fast-pacing ad popularity and network dynamism by reducing training times. The popularity of mobile games might change rapidly, and classification algorithms should consider fresh data that represent those changes. By implementing a two-stage training approach and by subsampling the negative class (Chapelle et al. 2015), we can shorten training times and therefore incorporate more frequent data updates. We first train our model using generic (publisher-independent) features; for each publisher, we retrain our model to include publisher-specific information (i.e., interaction terms). This approach allows us to perform more frequent data updates and helps when new publishers are added, because we use the publisher-independent model until we gather enough data to incorporate publisher-dependent information.
- Its software infrastructure uses an array of technologies, thus allowing us to retrieve, cleanse, and utilize data that account for more than 150 million ad requests per week, while we retrieve more than 40 days of user history data to train our models.

Our models also incorporate policy restrictions posed by advertisers. Before elaborating on the specificities of our approach, we provide a concise outline of related research.

Related Research

Research related to online and mobile advertising has proliferated in recent years, addressing a diverse range of topics. However, to the best of our knowledge, our work is the first to consider the dynamic allocation of video advertisements that are displayed within mobile applications (i.e., in-app ads). This context poses some unique characteristics that influence the design of the serving

algorithms and have not been addressed previously, such as the small time window during which an ad should be displayed, and the high user engagement with the host application.

Related research focuses on traditional advertising environments and problems, such as scheduling television commercials and sales plans (Bollapragada et al. 2002, Bollapragada et al. 2004, Popescu and Crama 2016) and scheduling promotional ad text messages, which are sent to customers in shopping centers (De Reyck and Degraeve 2003). Andrews et al. (2016) measure the effectiveness of ad text messages that are sent to consumers in crowded places. The authors demonstrate that, perhaps counterintuitively, being in a crowded subway train increases consumers' purchase rates, a behavior likely attributable to mobile immersion. Our problem is similar to this stream, because we allocate ads to users, and advertisers require their campaigns to be of specific duration. In our setting, however, users are more engaged with the host application and dynamically generate advertisement slots. As a result, the ad-allocation decisions must be made instantaneously.

Display advertising is an important category of computational advertising, where advertisers pay publishers for placing graphic ads (banners) on their Web pages. Online ad serving is a focal problem of computational advertising; examples of the topics studied include the allocation of combined sales and brand ads (Saeed et al. 2009), targeted display advertising (Perlich et al. 2014, Turner 2012, Hojjat et al. 2014), impression pricing (Cohen et al. 2016), capacity allocation of online ads (Araman and Fridgeirsdottir 2011), firm competition with respect to targeting individuals on social networks (Bimpikis et al. 2016), and online auctions (Muthukrishnan 2009, Balseiro et al. 2015). Although Web and mobile ads are both based on high-frequency ad-allocation algorithms, Web ad slots are generated in batches (e.g., when a user visits a Web page), while only one mobile ad slot is generated per user visit. Moreover, mobile video ads may interrupt user activity and degrade user experience to a greater degree than Web banners. Some recent studies

have focused on the use of click-through rates (CTRs), a focal performance metric in online advertising, and how they can be adjusted to better capture profitable user actions that are triggered by sequences of clicks (Wu et al. 2012, Xu et al. 2014, Besbes et al. 2016). Mobile advertising networks use a variety of simple CTR-related metrics, because users do not trigger subsequent actions; therefore, advertiser profits can be attributed to unique events. The recent work of Hao et al. (2016), who investigate optimal contract structures in in-app mobile advertising is also relevant.

From a methodological point of view, our work is closest to the machine-learning framework of Chapelle et al. (2015), in which the authors train logistic regression models that predict CTRs. Our models are also based on logistic regressions; however, we address two additional challenges: (1) ad allocations should conform to advertiser budgets, and (2) users might exhibit fatigue if they are exposed to the same ad multiple times, which is the defining attribute of in-app mobile advertising. We address these extra characteristics through an online heuristic that ensures the smooth consumption of ad inventories and utilizes a rotation allocation mechanism that considerably improves CTRs. Until now, little attention has been given to the problem of ad fatigue in mobile advertisements, despite evidence that consumers strongly feel these ads are more invasive than ads in other media (Upstream and YouGov 2012).

Vungle's Legacy Ad-Allocation Algorithm

Since its infancy, Vungle has strived to deliver the best possible customer experience. Responding efficiently to thousands of ad requests per second takes considerable effort; because Vungle's priority was to build a stable and reliable ad network, the first version of its ad-serving algorithm was robust but straightforward: given a publisher app, Vungle would calculate, for each advertiser, the probability of a revenue-generating user action and corresponding expected revenue, and would

then display the ad that will provide the most profit to the advertiser and to Vungle. That probability was based on the average response of all previous impressions of that ad. A significant advantage of this approach is its speed and robustness. However, it does not utilize the multitude of user-related data that are now available with each request, and which can significantly affect the likelihood of a user taking a revenue-generating action. Vungle subsequently implemented a more advanced ad-allocation algorithm, which included geo-targeting; that is, it takes into account the country from which each request originates, resulting in a probability of a profitable user action for each ad video, for each {publisher, country} pair. See Appendix A for a formal description of the legacy algorithm. This additional granularity significantly improved Vungle's eRPM, and we used this algorithm as the benchmark for our new model, which uses more user-specific data to increase the likelihood of a user taking a revenue-generating action.

Improving Estimations of Revenue-Generating Actions: A Machine-Learning Framework

The core component of an ad-serving algorithm is a classification model that computes, for each candidate ad video, the probability that it will generate a profitable action (i.e., a view, click, or installation. Recognizing that the serving algorithm should utilize accurate, and therefore sophisticated, classification models, but simultaneously run almost instantaneously, we adopted a two-stage approach; we trained and validated classification models, tested them offline, and entered them into a real-time algorithm that incorporates budget restrictions and user-fatigue issues.

Classifier Selection

For prediction accuracy, sophisticated ensemble methods, such as random forests, boosted trees, and bagged trees, tend to outperform other methods, such as support vector machines, uncalibrated neural networks, and logistic regression (Caruana and Niculescu-Mizil 2006, Wu et al. 2012). In

a production environment, however, a trade-off between prediction accuracy and speed of serving or robustness of implementation cannot be made. Although ensembles of trees (e.g., random forests) can create predictions in a reasonable time, the training time they require is usually higher than that of individually trained classifiers, and training them using datasets with hundreds of millions of cases could be time consuming. This is particularly important in an environment such as mobile advertising, where popularity trends change rapidly, because a model that does not use the most recent data can result in predictions of inferior quality. Therefore, to the best of our knowledge, the dissemination of ensemble methods into large-scale production systems is yet to realize the full potential of these methods, at least in the mobile advertising industry. Another important limitation of ensemble methods is that they are nontrivial to update incrementally; that is, models may have to be retrained and tested from scratch when new data are obtained. Building the prediction model anew as soon as possible requires considerable investment in software infrastructure, is hard and expensive to scale to larger volumes of data, and amplifies the impact of long training times.

Therefore, we opted for logistic regression methods, which have a straightforward probabilistic interpretation, are easier to interpret and communicate to management, are easy to parallelize, scale, and update, and can be adapted to incorporate exploration and exploitation strategies, such as Thompson sampling (Chapelle and Li 2011). The training of logistic regression models can be parallelized not only at the cross-validation phase, but also at the lower level of solving the associated optimization problem, with solving techniques such as parallelized stochastic gradient descent (Zinkevich et al. 2010). This parallelization leads to substantially faster training times, thereby allowing more frequent data updates. In addition, the Bayesian interpretation of logistic regression (Bishop 2006) allows incremental data updates; that is, the models do not need to be trained from scratch, but only to incorporate new data as soon as these data become available.

Finally, using logistic regression makes possible subsampling the negative class (i.e., cases in which no profitable user action results). In our application, this class forms 99.9 percent of the dataset, and therefore leads to substantial computational gains (Chapelle et al. 2015). These factors mean that logistic regression models are faster to train, and easier to update, maintain, scale, extend, and interpret, in contrast to more advanced methods, which typically require more time to train, more resources to maintain, and more software infrastructure to sustain and extend. For these reasons, we selected logistic regression as the core prediction model of our approach. At the time of this writing, the latest version of our system incorporates both incremental updates and Thompson sampling, although subsampling the negative class was part of our first implementation. See Appendix B for the details of our prediction model.

Data Diversity and Model Granularity

To predict the likelihood of a user taking a revenue-generating action, we utilize a multitude of user features that are part of each ad request. These include (1) user-specific features, such as language, device type, orientation, and sound level; (2) geographical and temporal features, such as time of the day, day of the week, country, time zone, and city; (3) advertisement-specific features, such as the length of time each campaign has been active and when it will conclude, and the number of views, clicks, and installations during the previous five days; and (4) interaction features, such as the number of times a user has seen each candidate ad and the last time that user saw the ad, which relate to possible user fatigue. In total, we used 40 features; see Appendix C for details.

Coping with User Fatigue: Service Randomization

Measuring and predicting user fatigue is complicated: showing an ad to the same user repeatedly might have both positive and negative effects. In television advertising, for example, repetition is

key to inducing consumers to buy a product or service (Krugman 1965); however, excessive repetition can have the opposite effect. Perhaps more importantly, displaying the same ad prevents the display of other candidate ads (i.e., ads from the same advertiser or from a competitor), which could have been more effective had they been aired with enough frequency to overcome the user's initial inertia.

To cope with these issues, we use a randomization algorithm, which dynamically alters the ranking of candidate ads by considering (1) a user's level of engagement with the publisher app, and (2) the user's reactions to recently displayed ads. Our intention is to more rapidly alter the ranking of candidate ads when a user generates ad requests at a high rate, and to become less aggressive in altering the ranking when a user generates ad requests at a lower rate. Specifically, our algorithm displays the ad with the highest eRPM with probability 1-p, and applies the following procedure with probability p, which we set equal to 70 percent after experimentation: For each candidate video, we record the elapsed time (in minutes) that has passed since it was last displayed to that user. If the video has been recently displayed to that user, we push it to the bottom of the candidate-video list; otherwise, we calculate its corresponding eRPM. A video is classified as recently displayed if its recency score is less than the number of candidate videos. The recency score is a strictly increasing affine function of each video's elapsed time. Contrary to other randomization algorithms that randomize based on relative scores (e.g., randomized first choice), our algorithm achieves randomization by adapting to user reactions. Appendix B shows an example that includes three candidate videos.

Note that the longer the list of candidate videos, the more likely that the top videos rotate, therefore diversifying the selection process. Also, when a user is very engaged with the host mobile application and generates frequent requests, the result is a low recency score for each previously

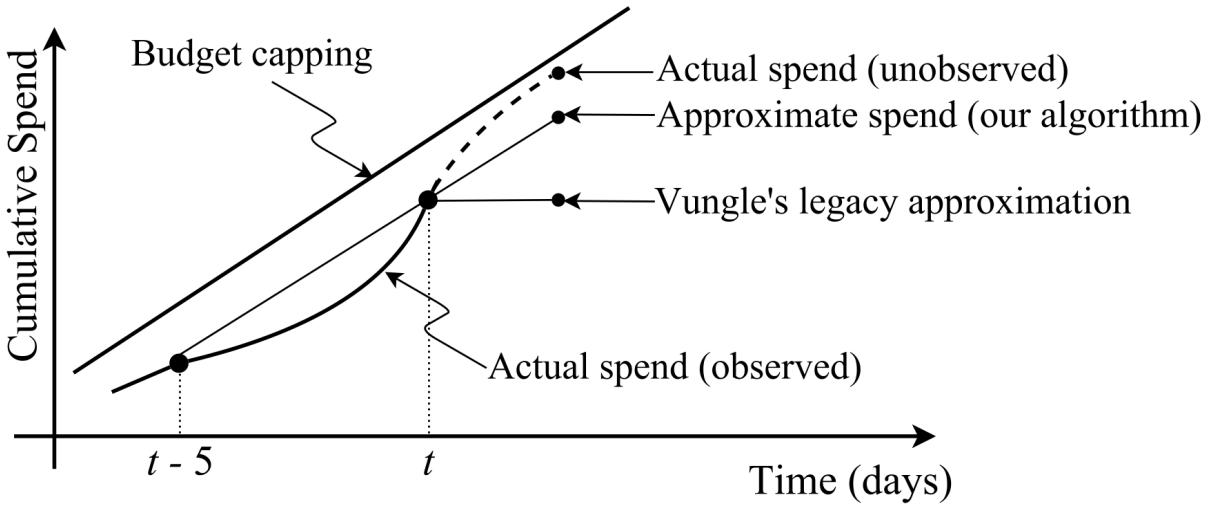
displayed ad, excluding it from immediate selection. Although Vungle does not control the frequency with which ads requests are generated, we have found that our rule strikes a good balance between displaying high-eRPM ads when user engagement is moderate and diversifying the selected ads when user engagement is high. In extreme cases in which a user has rotated the entire list of candidate ads, we stop serving ad requests until the recency scores improve. This process has proved helpful in maintaining the good experience of users who are highly engaged with their publisher app.

Incorporating Advertiser Restrictions: Budget Capping

Budget or frequency capping (i.e., restricting the number of times a video ad is displayed) is commonplace in online advertising. Companies use it to avoid user burnout, and to ensure that a campaign's budget is not consumed too fast. For example, an advertiser might initiate a $1 million campaign over two months, with a daily cap of $25,000. Although advertisers control the capping requirements, Vungle is responsible for their implementation.

A challenge specific to mobile advertising is that some pricing models exhibit uncertainty on whether the user has generated a profitable action. In pay-per-install pricing, the predominant pricing model, the advertiser is billed only when the user downloads and opens the corresponding app. Thus, at any given moment, there are users who have downloaded but not yet opened the app. To address this, we calculate the expected budget spent during the past hour by prorating the average budget spend rate of the past five days, which is the most recently available data. We made our decision to calculate the expected budget spent during the previous hour because more than 90 percent of installs occur one hour after the user has watched the corresponding video. Despite its simplicity, our approach results in a more accurate representation of the unobserved budget spent when we compare it with Vungle's legacy policy (Figure 2).

Figure 2. Our Algorithm Approximates The Cumulative Spend During The Past Hour To Determine If Capping Should Be Applied



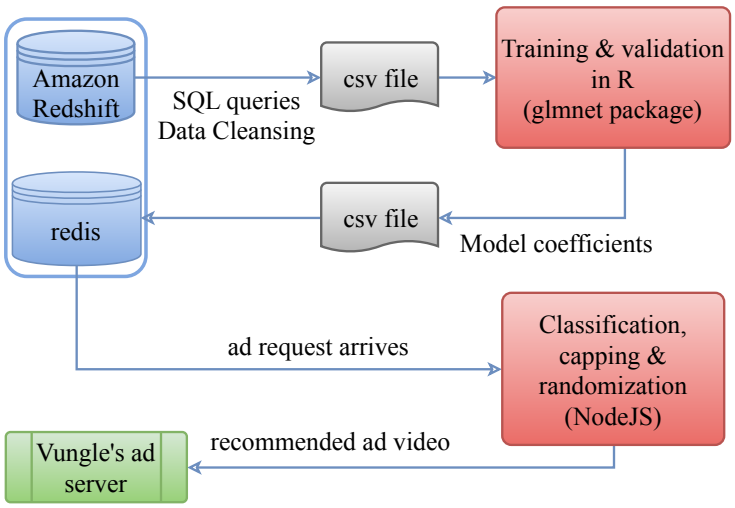
New Customers and New Users

Vungle's customer base includes approximately 250 million unique users and 25,000 publishers and advertisers, with thousands of new users acquired each day. This poses challenges to the prediction models, because no data are available on these new users. Incorporating new advertisers is straightforward, because the only advertiser-dependent feature in the model is the number of times a user has seen a given ad (i.e., zero for new advertisers). Considering new publishers, however, is not straightforward, because the logistic regressions include a set of parameters based on interactions between publishers. Therefore, we use publisher-independent features to determine install probabilities for a publisher if the number of ad slots that publisher has generated is less than 5,000; otherwise, we train a new model that includes publisher-specific information. Finally, our model includes features that depict user history, such as user installations, the number of ads that a user has seen, and the corresponding timestamps of these ads. The model then predicts the "average" user behavior, which is updated as the user takes specific actions (e.g., views and installs).

Implementation and Adoption

To ensure the scalability and robustness of our model, we used an array of diverse technologies that (1) retrieve and cleanse the data; (2) train and validate the models; and (3) implement the frequency capping and randomization components. Figure 3 depicts a high-level representation of the ad-serving architecture we adopted.

Figure 3. Our Model Has Three Components: Data Cleansing (SQL Queries), Model Training (In R), And Algorithm Implementation (In NodeJS)



First, we gathered data from Vungle's data warehouse (Amazon's Redshift) using SQL queries. The data-collection horizon is crucial, because it affects the predictive power of the classification models. We utilized recent big-volume data to ensure that (1) we adequately capture the popularity of each video, and (2) a substantial combination of features is present. To this end, we used data from a prior one-week period, which accounted for more than 150 million ad requests. Specifically, we used one week of ad-request data to train the model; for each ad request in that

training set, we considered 40 days of user-history data. The collection procedure was repeated daily.

Following the data-collection step, we trained logistic regression models using the R programming language and the glmnet library (Friedman et al. 2010). To ensure model parsimony and good out-of-sample performance, we tested alternative configurations of the elastic-net regularization method, which is a generalization of LASSO and ridge regression (Zou and Hastie 2005, Hastie et al. 2009). We decided to use LASSO regularization to impose some sparsity on the estimated coefficients to improve the interpretability of the models (because LASSO tends to drive less-significant coefficients values to zero). To cope with the computational burden of training our model, we adopted a two-stage approach; we first trained the first-order features, and then kept them fixed to train the interaction features. In addition, like Chapelle et al. (2015) and Breslow and Cain (1988), we subsampled the negative class (i.e., no user actions), which allowed us to reduce the training time of our models by an order of magnitude. Appendix D provides details.

When we completed the training phase, we uploaded the model coefficients on Redis, a key-value database (Redislabs 2017). The prediction, randomization, and frequency-capping procedures are implemented in JavaScript using the Node.JS framework (Surhone et al. 2010). We made this choice because Node.JS exhibits good network scalability and is compatible with Vungle's code base, therefore enabling Vungle's engineers to maintain, enhance, and adapt our algorithm to new business rules.

Adoption

To test the performance of the algorithm, Vungle routed about one-sixteenth of its ad requests, an average of 1.3 million user views per day, to our new algorithm in the spring of 2014. After a successful test, Vungle rolled out the new algorithm in the fall of 2014 and is still using it. Although the company is continually modifying the algorithm, it has retained the core aspects.

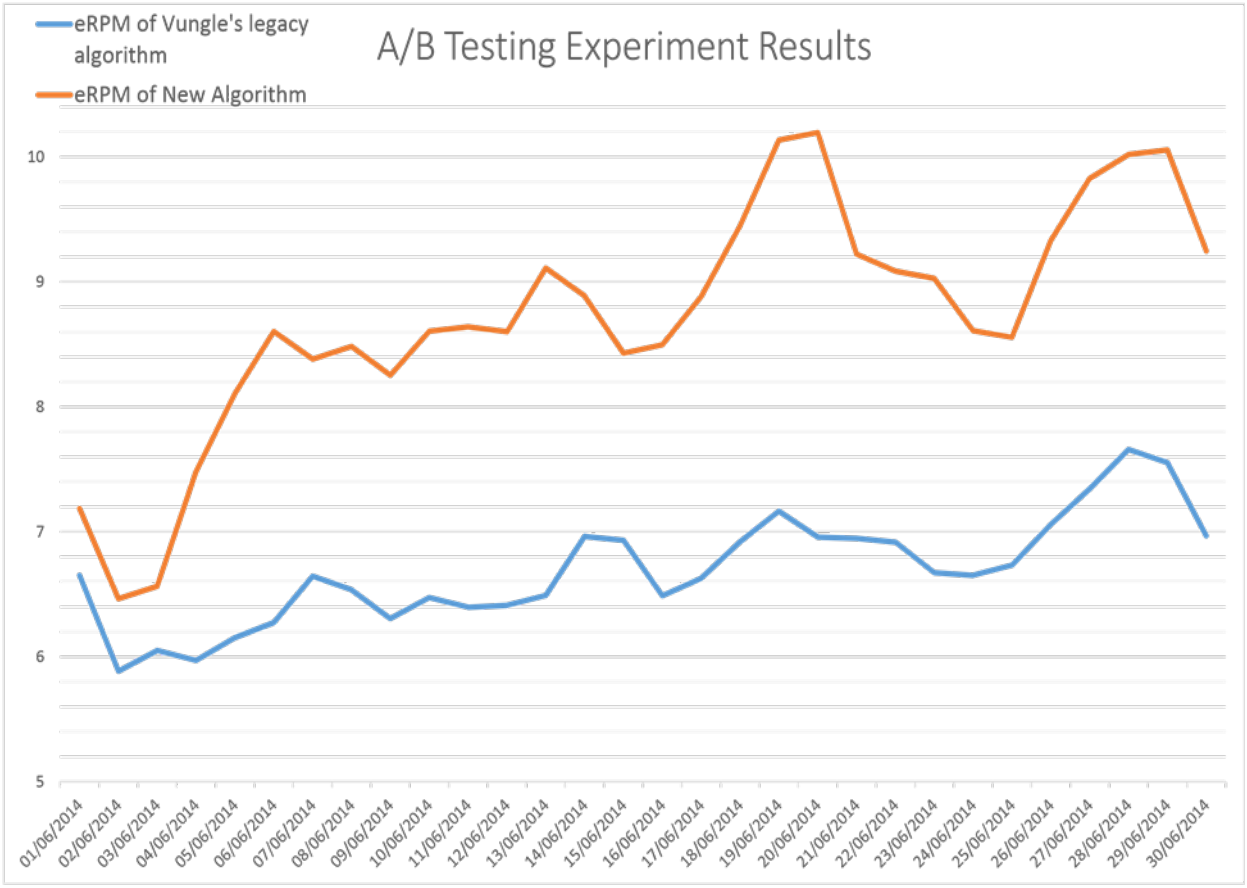
Benchmarking Performance: Benefits and Insights

A/B Testing

To assess the efficacy of the new algorithm, we performed several A/B tests to compare the algorithm's performance with that of Vungle's legacy algorithm. Figures 4 and 5 compare the performance of our algorithm with the performance of Vungle's best legacy algorithm for one A/B test that was performed in spring 2014. Figure 6 shows a detailed hourly overview and the corresponding eRPM. The results indicate a consistent 20 percent eRPM lift across all randomized buckets of users.

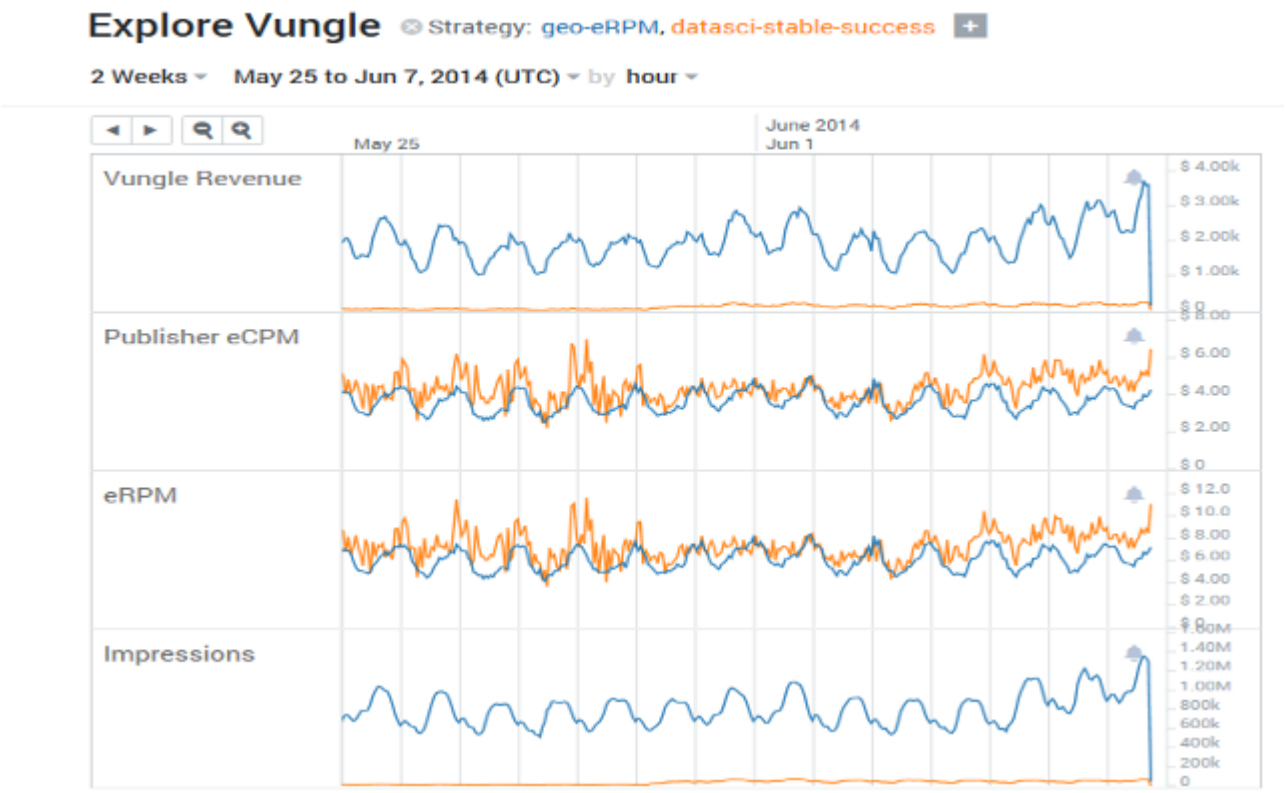
Figure 4. Our Algorithm (Top Line) Consistently Outperformed Vungle's Best Legacy Algorithm (Bottom Line) In Terms Of Expected Revenue Per Mille (eRPM)

Figure 5. A Second A/B Test, Which We Conducted In June 2014, Shows The eRPM Evolution Of Our Algorithm And Vungle's Legacy Algorithm



Note. Our algorithm (top line) outperforms Vungle's legacy algorithm (bottom line) by 30 percent.

Figure 6: Our Algorithm Outperforms Vungle's Legacy Algorithm With Respect To Publisher eCPM And eRPM



Note. The smaller number of impressions allocated to our algorithm (bottom graph, Impressions, bottom line) accounts for its more volatile behavior (middle graphs, jagged lines) and for the revenue difference (top graph, bottom line).

Insights and Follow-Up Projects

The detailed data exploration and analyses that were required for us to develop our new algorithm revealed several key managerial insights. They allowed us to detect the effects of higher-order interactions among features that have significant predictive power. For example, the probability of a user responding to an ad video has a positive linear relationship with the volume level of the

device. Although this seems intuitive in hindsight, it was not a factor that mobile advertising companies were considering at the time, principally because there is no ability to modify the sound volume; however, this inability (by advertisers) to modify the volume level might imply that advertising companies were not paying attention to the levels of sound-volume data. However, a closer analysis revealed that some ads were strongly affected by sound volume, whereas others were not; once Vungle knew the sound volume, it could display the best ad for each volume setting; Appendix E shows an example. This and similar insights led Vungle to systematically characterize the video contents, and to then seek other configurations that exhibit predictive power.

At the time of this writing, Vungle has set up an independent data science team, which has established a robust algorithmic environment and a stable testing environment. Moreover, it has introduced several video features, which interact with publisher apps, that exhibit significant predictive power. This has led to refining the prediction algorithm to the video creative level (i.e., generating estimates for specific variants of videos, called creatives, and capturing individual characteristics of each video creative, thus leading to predictions that are more powerful). Perhaps unexpectedly, having data on the predictive power of videos allows Vungle to determine the features that are most relevant for each user, and therefore contributes to the design of new video creatives. Vungle's creative designers now combine their creative stimuli with insights from data to improve the appeal of their video creatives.

Discussion

Although the advent of big data has brought numerous opportunities to improve the service level and economic efficiency of firms, their incorporation into decision-making procedures is yet to mature (Bertsimas and Kallus 2014). Our work suggests that the use of machine-learning models in mobile advertising, a relatively new field, can lead to fruitful research and to applications that provide significant impacts and important insights.

We believe that the key drivers of the success of our project are (1) the enhanced predictive ability of our classification algorithms, and (2) the incorporation of budget-capping requirements and user fatigue in real time. The implementation of these components led to a model that significantly improved eRPMs and increased monthly revenues by more than $1million. In addition, Vungle engineers have extended and enhanced our algorithms with additional features, which the firm's video designers also use when they launch new video creatives. This has led Vungle to embrace the value of scientific modeling, and to be confident that it is providing the best quality service to its customers and users.

Appendix A: Vungle's Legacy Algorithm

Figure A1. Below, We Display Vungle's Legacy Algorithm

Algorithm 1 Vungle's legacy algorithm

Input: $Ads, Publisher, Country$
Output: Ad
  $eRPM \leftarrow 0$
  for $a \in Ads$ **do** ▷ Average eRPM for this (publisher, country) pair
    $a.eRPM \leftarrow \frac{a.Revenue(Publisher,Country)}{1000}$
    if $a.eRPM > eRPM$ **then**
      $eRPM \leftarrow a.eRPM$
      $Ad \leftarrow a$
    end if
    Return Ad
  end for

Note. Vungle's legacy algorithm considers the average *e*RPM of each advertiser in each publisher-country pair and returns the *e*RPM-maximizing ad. For new advertisers, their *e*RPM estimates default to the publishers' estimates with whom they form pairs.

Appendix B: Logistic Regression and Service Randomization

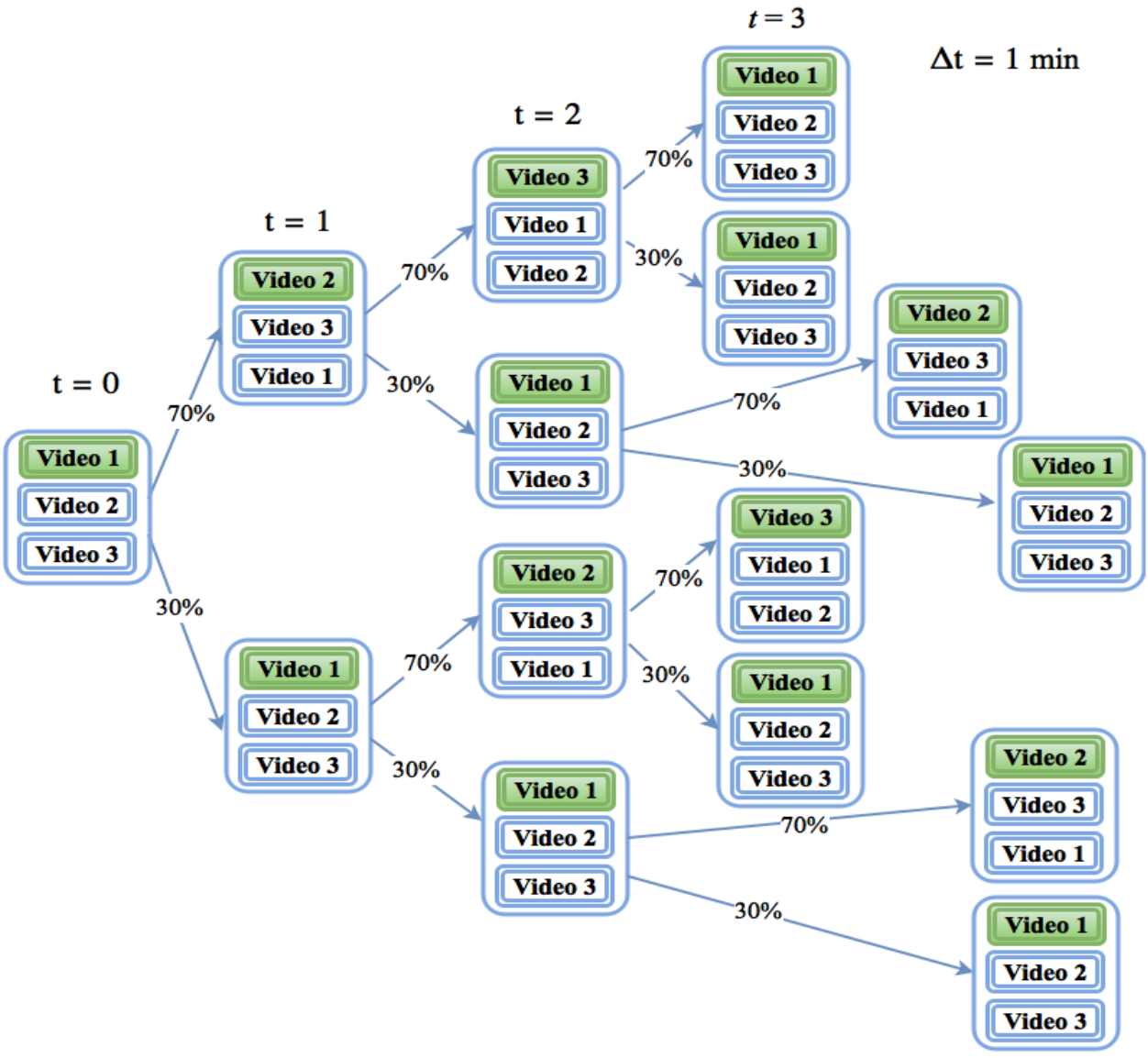
Our logistic regression classifier takes the form $\mathrm{E}[Y_i = 1 \mid \mathbf{X}_i] = \frac{1}{1+e^{-\beta \mathbf{X}_i}}$, where $\beta = [\beta_1,...,\beta_n]^T$ is the vector of coefficients and $\mathbf{X}_i = [x_{i1},...,x_{in}]$ is the vector of features for case *i*, including the constant term and the interaction terms with publisher. Large positive components of β increase the probability of the positive class (i.e., the probability of an installation). We calculate the eRPM of each ad using $p\mathrm{E}[Y_i = 1 \mid \mathbf{X}_i]$, where *p* is the price the advertiser pays per installation.

Service Randomization

Figure B1 shows an example of how the service randomization heuristic works when three candidate ad videos exist. In this case, we assume that Video 1 has the highest eRPM, followed by Videos 2 and 3, respectively. The recency function is $f(\Delta t_i) = \alpha\ \Delta t_i + \beta$ with $\alpha = 1$ and $\beta = 0$.

Therefore, Video 1 will be blocked with 70 percent probability if it has not been displayed during the previous three minutes, but its eRPM will be reactivated as soon as three or more minutes have elapsed since it was last selected. The heuristic is activated only for ads that the use has previously seen.

Figure B1. The Service Randomization Heuristic Produces A Video Recommendation When A User Generates Four Requests, One Minute Apart From Each Other



Note. The video selected in each scenario is placed on top of the list. Video 1 has the highest eRPM, followed by Video 2 and Video 3. A linear recency score function implies that when Video

1 has not been displayed for three minutes (which is the length of the candidate list), its eRPM is reactivated and it returns to the top of the candidate list.

Appendix C: Feature Description

Table C1 describes the features we utilized. Note that each feature was utilized both individually and at the time that it interacts with each publisher app. The number of user installs and ad views is discretized using breakpoint sets {1, 2, 5} and {1, 2, 5, 10}, respectively, which were determined via preliminary analysis (i.e., visualizations and hypothesis tests). We found that, ceteris paribus, users with a large number of user installs (i.e., 40 or more) do not seem to exhibit different behavior than users with 5 or more installs. This motivated us to discretize this feature, as opposed to adopting other alternatives, such as, introducing higher-order terms.

Table C1. We Use A Multitude Of Logistic Regression Features That Allow Us To Capture Refined Information About Each Ad Request

Feature Name	Category	Description
`Country`	Discrete categorical	Country of request origin
`device_make`	Discrete categorical	Device brand
`device_connection`	Discrete categorical	Device connection (WiFi, 3G, etc.)
`left1os`	Discrete categorical	Distinguishes iOS devices
`Dowviewed`	Discrete categorical	Day of week the request was sent
`Hourviewed`	Discrete categorical	Date range in which the request falls (`morning`, `midday`, `afternoon`, etc).
`device_volume_disabled`	Discrete categorical	=1 if device is muted, 0 otherwise
`Isipod`	Discrete categorical	=1 if device is an ipod
`Isipad`	Discrete categorical	=1 if device is an ipad
`Isportrait`	Discrete categorical	= 1 if device is in portrait orientation
`Intercept`	Discrete categorical	Constant term
`user_installs_1`	Discrete categorical	=1 if user has viewed and installed at least one app, 0 otherwise
`user_installs_2`	Discrete categorical	Similar to `user_installs_1`
`user_installs_5`	Discrete categorical	Similar to `user_installs_1`
`user_ad_views_1`	Discrete categorical	=1 if user has viewed the add before, 0 otherwise.
`user_ad_views_2`	Discrete categorical	Similar to `user_ad_views_1`

user_ad_views_5	Discrete categorical	Similar to **user_ad_views_1**
user_ad_views_10	Discrete categorical	Similar to **user_ad_views_1**
Adviewedpenalty	Discrete categorical	Penalty added if user has watched that particular add before (sample levels: first_time_watched, less_than_5_min, less_than_2_days, etc).
Anyadviewedpenalty	Continuous	Penalty added if user has watched an ad before, defined similar to recentinstallbonus (see below)
Recentinstallbonus	Continuous	Bonus added if user has installed recently an app, = $\exp(-\Delta t / a)$
device_volume	Continuous	Device sound level, normalized

Note. We modified some data in the table for confidentiality. In "recentinstallbonus," $\boldsymbol{\Delta t}$ represents the time difference between the request and when the user last installed an app. The positive constant $\boldsymbol{a}$ is kept confidential.

Appendix D: Implementation Details: Model Training

To find the logistic regression coefficients, we solve the following optimization problem:

$$\min_{\beta \in \mathbb{R}^n} \{\|y - \mathbf{X}\beta\|_2^2 + \lambda_1 \|\beta_1\|_1 + \lambda_2 \|\beta_2\|_2^2\}. \tag{D1}$$

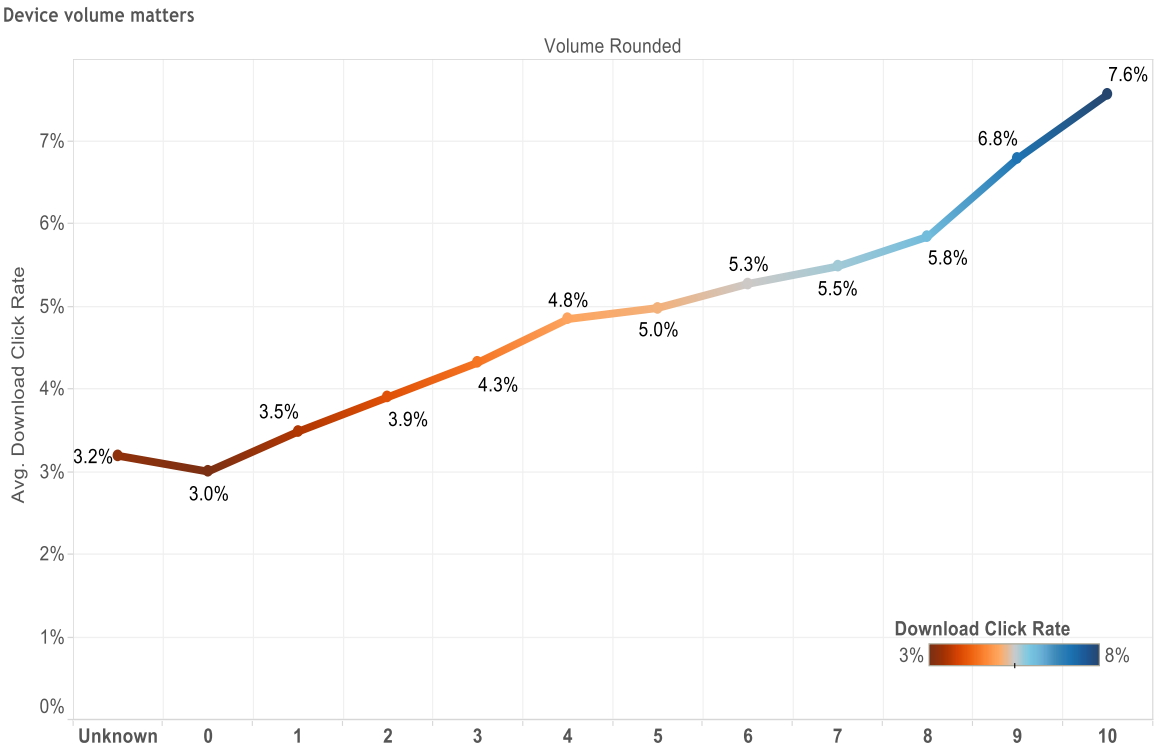
Zou and Hastie (2005) developed elastic net regression, (D1), to combine the benefits of the LASSO and RIDGE models, which use the one and two norms, respectively. In our computational experiments, LASSO balanced stable numerical behavior (i.e., convergence to the optimal solution) and good performance; therefore, we adopted it. It also imposes sparsity on the estimated coefficients, and therefore improves the interpretability of the models (since it tends to drive less-significant coefficients values to zero). By adopting a two-stage approach, we heuristically solve this problem. First, we solve a variant of (D1); we do not consider any interaction with publishers. Thus, the size of vector β equals the number of features (i.e., 40). Once we solve this optimization problem, we fix these values of β and solve a new optimization problem for each publisher. The new optimization problems (one for each publisher) are the same size as the previous generic

problem. To reduce the computation time, we subsample the negative class (i.e., no installation) at a rate of 2 percent. By using subsampling, we can adjust each resulting coefficient, including the intercept (Donkers et al. 2003, Chapelle et al. 2015). This technique reduced the computation time by an order of magnitude, while the corresponding accuracy loss was negligible.

Appendix E: Feature Visualization

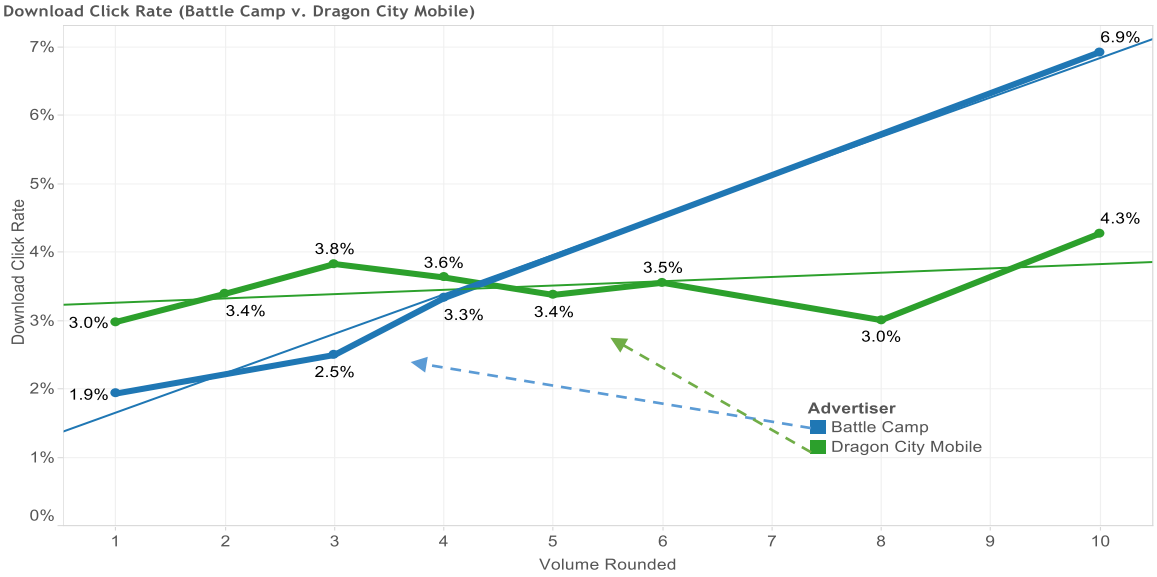
In this section, we present the visualization of some important features, and their interactions.

Figure E1. The Device Sound Volume Is Normalized From 0 To 10, While The Average Download Click Rate (Download Probability) Varies Between 3 Percent And 7.6 Percent



Note. The download probability improves as the device sound increases.

Figure E2. Device Sound Volume (X Axis, Measured From 1 To 10) Has A Strong Effect On The Downloaded Click Rate (Y Axis) For Some Videos, But Not For Others



Note. On closer examination, we found that videos with narratives (e.g., Battle Camp) are sensitive to the sound level, while videos with music only (e.g., Dragon City Mobile) are not.

Figure E3. Time of The Day (Horizontal Axis) And Location (Here: London) Have A Strong Interaction Effect, On Both The Number Of Views (Vertical Axis) And On The Download Probability (Download Click Rate, Indicated By The Percentage Labels Along The Line)

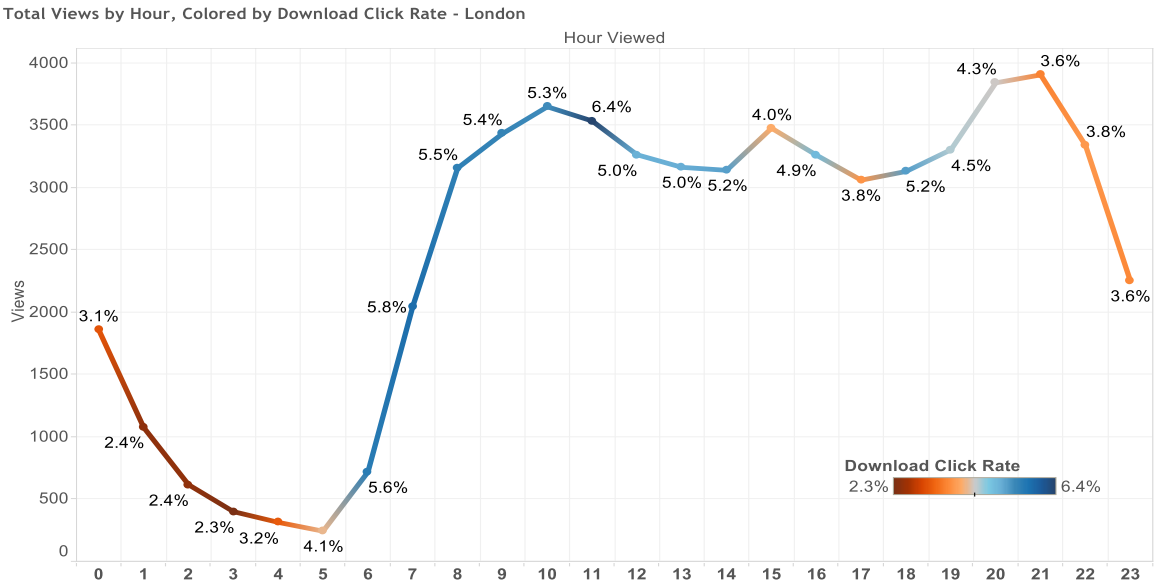
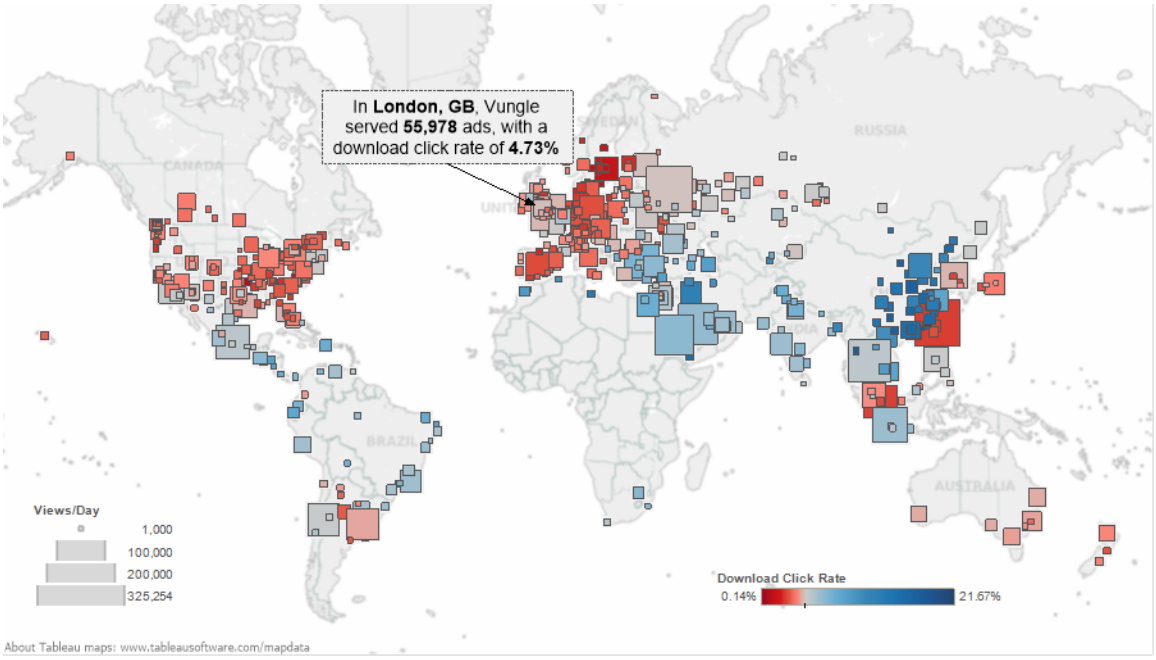


Figure E4. Location Appears To Be A Differentiating Feature Relative The Average Number Of Daily Views (Size Of Each Square), And More Importantly To The Probability Of A Download (Download Click Rate)



Note. As an example, the large dark squares in Emirates indicate a large number of views and a high download click rate. The small dark squares across the United States indicate a low number of views and a low download click rate. The dark square in China indicates a very high download click rate, while the dark square in Hong Kong indicates a very low download click rate.

Acknowledgments

We acknowledge the contributions and support of Vungle engineers, without whom this project would not have been possible. We are especially grateful to Vungle's founder and CEO, Zain Jaffer, for his leadership and strong support throughout the project.

References

Araman VF, Fridgeirdottir K (2011) Cost-per-impression pricing and campaign delivery for online display advertising. Working paper, Olayan School of Business, Beirut, LB.

Balseiro SR, Besbes O, Weintraub GY (2015) Repeated auctions with budgets in ad exchanges: Approximations and design. *Management Sci.* 61(4):864–884.

Bertsimas D, Kallus N (2014) From predictive to prescriptive analytics. Accessed January 19, 2017, https://pdfs.semanticscholar.org/b3ff/4b36c4e24d1cbf54777140e4a69b666b3e51.pdf.

Besbes O, Gur Y, Zeevi A (2016) Optimization in online content recommendation services: Beyond click-through rates. *Manufacturing Service Oper. Management* 18(1):15–33.

Bimpikis, K, Ozdaglar A, Yildiz E (2016) Competitive targeted advertising over networks. *Oper. Res.* 64(3):705–720.

Bollapragada S, Cheng H, Phillips M, Scholes M, Gibbs M, Humpherville M (2002) NBC's optimization systems increase its revenues and productivity. *Interfaces* 32(1):47–60.

Bollapragada S, Bussieck MR, Mallik S (2004) Scheduling commercial videotapes in broadcast television. *Oper. Res.* 52(5):679–689.

Breslow N, Cain KC (1988) Logistic regression for two-stage case-control data. *Biometrika* 75(1):11–20.

Caruana R, Niculescu-Mizil A (2006) An empirical comparison of supervised learning algorithms. *Proc. 23rd Internat. Conf. Machine Learning* (ACM, New York), 161–168.

Chapelle O, Manavoglu E, Rosales R (2015) Simple and scalable response prediction for display advertising. *ACM Trans. Intelligent Systems Tech.* 5(4):1–34.

Chapelle O, Li L (2011) An empirical evaluation of Thompson sampling. Accessed January 19, 2017, http://www.research.rutgers.edu/~lihong/pub/Chapelle12Empirical.pdf.

Cohen MC, Lobel I, Paes Leme R (2016) Feature-based dynamic pricing. Working paper, New York University, Stern School of Business, New York.

De Reyck B, Degraeve Z (2003) Broadcast scheduling for mobile advertising. *Oper. Res.* 51(4):509–517.

Donkers B, Franses P, Verhoef P (2003) Selective sampling for binary choice models. *J. Marketing Res.* 40(4):492–497

Friedman J, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. *J. Statist. Software* 33(1):1–22

Gandhi V (2015) An insight into the US mobile advertising market: From rich media to native ads. Report, Frost & Sullivan, San Antonio, TX.

Guerin H (2016) Data provided via email from Hammond Guerin to Ioannis Fragkos, April 28, 2016.

Ha A (2014) In-app video ad startup Vungle raises $17M more. Accessed June 1, 2016, http://techcrunch.com/2014/02/06/vungle-series-b/.

Hastie TR, Tibshirani R, Friedman J (2009) *The Elements of Statistical Learning* (Springer Series in Statistics, Springer, New York).

Hojjat A, Turner J, Cettintas S, Yang J (2014) Delivering guaranteed display ads under reach and frequency requirements. *Proc. Twenty-Eighth AAAI Conf. Artificial Intelligence* (AAAI, Palo Alto, CA), 2278–2284.

Krugman HE (1965) The impact of television advertising: Learning without involvement. *Public Opinion Quart.* 29(3):349–356.

Muthukrishnan S (2009) Ad exchanges: Research issues. Leonardi S, ed. *Internet and Network Economics* (Springer, Berlin Heidelberg), 1–12.

Perlich C, Dalessandro B, Raeder T, Stitelman I, Provost F (2014) Machine learning for targeted display advertising: Transfer learning in action. *Machine Learn.* 95(1):103–127.

Popescu DG, Crama P (2016) Ad revenue optimization in live broadcasting. *Management Sci.* 62(4):1145–1164.

Redislabs (2017) Redis multi-model infinite uses. Accessed March 13, 2017, https://redis-labs.com/.

Saeed A, Arcaute E, Khuller S, Ma W, Malekian A, Tomlin J (2009) Online allocation of display advertisements subject to advanced sales contracts. Accessed January 19, 2017, http://www.cs.cornell.edu/~saeed/archive/online_ad.pdf.

Surhone LM, Tennoe MT, Henssonow SF (2010) *Node.JS* (Betascript Publishing, MU).

Turner J (2012) The planning of guaranteed targeted display advertising. *Oper. Res.* 60(1):18–33.

Wu KW, Ferng CS, Ho CH, Liang AC, Huang CH, Shen WY, Jiang JY et al. (2012) A two-stage ensemble of diverse models for advertisement ranking in KDD Cup 2012. Accessed January 19, 2017, https://kaggle2.blob.core.windows.net/competitions/kddcup2012/2748/media/NTU.pdf.

Xu L, Duan JA, Whinston A (2014) Path to purchase: A mutually exciting point process model for online advertising and conversion. *Management Sci.* 60(6):1392–1412.

Upstream and YouGov (2012) 2012 Digital advertising attitudes report. Accessed June 1, 2016, http://cache-www.upstreamsystems.com/wp-content/uploads/2014/02/yougov12_report.pdf.

Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. Accessed January 19, 2017, http://users.stat.umn.edu/~zouxx019/Papers/elasticnet.pdf.