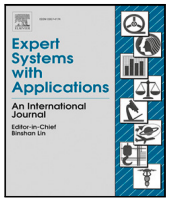
Expert Systems With Applications 231 (2023) 120659

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



A novel evaluation framework for recommender systems in big data environments

Roberto Henriques [a,*], Luís Pinto [b]

[a] NOVA IMS Information Management School, Universidade Nova de Lisboa, 1070-312, Lisboa, Portugal
[b] Instituto de Investigação e Formação Avançada (IIFA), University of Évora, Largo dos Colegiais 2, Évora, 7000, Portugal

ARTICLE INFO

Keywords:
Recommender systems
Mobile app store
Evaluation metric
Big data environment

ABSTRACT

Recommender systems were first introduced to solve information overload problems in enterprises. Over the last few decades, recommender systems have found applications in several major websites related to e-commerce, music and video streaming, travel and movie sites, social media, and mobile app stores. Several methods have been proposed over the years to build recommender systems. However, very little work has been done in recommender system evaluation metrics. The most common approach to measuring recommender system's performance in offline settings is to employ micro or macro averaged versions of standard machine-learning measures. Profit or other business-oriented metrics have been proposed for other predictive analytics problems, such as churn prediction. However, no such metrics have emerged for the recommender system context. In this work, we propose a novel evaluation metric that incorporates information from the online-platform userbase's behavior. This metric's rationale is that the recommender system ought to improve customers' repeated use of an online platform beyond the baseline level (i.e. in the absence of a recommender system). An empirical application of this novel metric is also presented in a real-world mobile app store, which integrates the dynamics of large-scale big data environments, which are common deployment scenarios for these types of recommender systems. The resulting profit metric is shown to correlate with the existing metrics while also being capable of integrating cost information, thereby providing an additional business benefit context, which allows us to differentiate between two similarly performing models.

1. Introduction

It is often claimed that the internet changed retail businesses. For the first time, retailers were not limited to an assortment of popular items and could profit from endless product variety (Brynjolfsson, Hu, & Smith, 2006). This variety implies that the aggregated demand for niche products is comparable to the top most popular products – a phenomenon known as the "long tail" effect (Anderson, 2008). Two main factors are usually suggested as the cause of this effect (Goel, Broder, Gabrilovich, & Pang, 2010): one related to the supply side (retailers/producers) and another related to the demand side (consumers).

On the supply side, online retailers can include a vast number of items on their assortment when compared to traditional brick and mortar retailers (Brynjolfsson et al., 2006). This fact would theoretically be an advantage because demand is heterogeneous. Therefore a larger number of items would allow the retailer to provide more utility to a more significant number of customers (Quan & Williams, 2017). Some empirical evidence of this fact has been found (Goel et al., 2010), suggesting that all consumers have a small portion of niche products in their choices. Also, in the context of digital-goods distribution (which can include products such as apps, music, video and written content), advances in cloud computing allowed for almost linear scaling of data storage infrastructure, which is necessary to attain vast assortments with "long-tail" properties (Weinhardt et al., 2009).

On the consumer side, information search costs are hypothesized to be lower for consumers in the online context while also intensifying sellers' price competition (Choi & Mela, 2019). With provided search engine and recommender system capabilities, users can access the relevant content from the assortment (Schnabel, Bennett, & Joachims, 2018). In the particular case of mobile app stores, search engines have increasingly key importance, as evidenced by the rising importance of app store optimization (ASO) for publishers in mobile app stores (Wilson, 2018). ASO refers to the tactics employed to improve app stores visibility, similar to search engine optimization (SEO) (Bilgihan, Kandampully, & Zhang, 2016). This appears to confirm that these systems do play a role in lowering information search costs for consumers.

* Corresponding author.
E-mail addresses: roberto@novaims.unl.pt (R. Henriques), d41900@alunos.uevora.pt (L. Pinto).

https://doi.org/10.1016/j.eswa.2023.120659
Received 19 April 2021; Received in revised form 23 February 2023; Accepted 30 May 2023
Available online 14 June 2023

Several studies have provided evidence that recommender systems positively impact content discovery in online environments (Brynjolfsson, Hu, & Simester, 2011; Fleder & Hosanagar, 2007; Pathak, Garfinkel, Gopal, Venkatesan, & Yin, 2010), which should reinforce the "long tail".

Although the long tail effect's impact of the "long tail" effect is usually considered very important for businesses, empirical evidence suggests that this effect does not occur in every digital market. In markets such as the online music industry, demand tends to follow a "superstar" effect (Rosen, 1981), also referred to as the "winner-takes-all" (Frank Robert & Cook Philip, 1995) or "blockbuster" (Fleder & Hosanagar, 2009) effect. In these markets, the most popular products' demand largely exceeds the aggregated demand for the least popular ones. The mobile app economy has been found to behave similarly (Zhong & Michahelles, 2013). This "superstar" effect is consistent with marketing science because it can be explained by two other empirical generalizations that tend to occur across markets (Zhong & Michahelles, 2012, 2013): the natural monopoly effect and the double jeopardy law (Ehrenberg & Goodhardt, 2002; Ehrenberg, Goodhardt, & Barwise, 1990; Ehrenberg, Uncles, & Goodhardt, 2004). It has been argued that management information systems such as search engines and recommender systems are especially relevant for these markets (Yin, Cui, Li, Yao, & Chen, 2012; Zhong & Michahelles, 2013), as long as they are able to increase sales diversity.

Some authors argue that different recommendation methods yield different impact levels in search cost reduction and therefore on sales diversity (Fleder & Hosanagar, 2007, 2009; Zhong & Michahelles, 2013). This difference suggests that long tail effects might be achieved in superstar markets by having more intelligent recommender systems or by avoiding the usual methods based on collaborative filtering, which tend to reinforce the most popular items (Peltier & Moreau, 2012; Zhong & Michahelles, 2013).

Given that firms have limited resources to allocate to machine learning projects, it follows that a recommender system's efficiency should be measured using an economic profit metric, in order to facilitate project prioritization according to business value (Jannach & Jugovac, 2019). This metric should incorporate the estimated financial impact of improvements in customer experience. The improvement should be measured against a baseline, which can be estimated from natural monopoly and double jeopardy effects. Double jeopardy effects are regularly observed in online retail and e-commerce (Huang, 2011), and these appear to extend to m-commerce settings, in particular, the mobile app market, as we have already seen (Zhong & Michahelles, 2013). Since these effects explain mobile app choice behavior and e-commerce in general, we can assume it explains mobile app store choice (at least in the Android platform), with public data confirming that a significant portion of Android users uses third-party app stores other than Google Play (App Annie, 2017). Finally, the metric should also incorporate a holistic cost accounting of the model training, validation, testing and deployment at Big Data scale, including cloud computing costs.

Our goal is to introduce a novel recommender system evaluation framework that combines both business benefits and technology costs and can be used to compare different models in an offline setting. This metric integrates information about the online platform user behavior, which allows it to become a business-aware measure of model performance.

This work has both technical and managerial impact. On the technical side, we provide an offline machine learning metric that can be adapted to a wide variety of industrial applications. On the managerial side, we hope to provide a theoretical framework to evaluate recommender systems that can be easily understood by managers but can also provide insights about user behavior in online platforms, similar to existing work being done in recommender systems (Ansari, Li, & Zhang, 2018; Ghoshal, Kumar, & Mookerjee, 2015).

In the following sections, we will present a literature review covering Recommender System Evaluation, and we will introduce our Action Value Estimator. Finally, we will present the main empirical results, followed by the conclusions, including a discussion of the results and their implications, the limitations of the study and possible directions for future research.

2. Recommender system evaluation

Model evaluation in the context of recommender systems includes several different metrics, which range from multi-class/multilabel versions of traditional machine learning and data mining ones to specific profit-centric indicators (Gilotte, Calauzènes, Nedelec, Abraham, & Dollé, 2018; Ju, Choi, Kim, & Moon, 2017). We can divide the evaluation metrics into three main groups: offline, online, and user studies (Beel & Langer, 2015).

2.1. Offline evaluation metrics

Offline evaluation metrics are the most common and assume that models can be evaluated in terms of prediction/classification accuracy of past item consumption data or ratings. In this context, if an item is recommended to a user who has not previously consumed it, we count it as a model failure. It is easy to see that offline metrics are not guaranteed to be realistic since the fact that an item was not previously consumed or rated does not mean it is not relevant to that user.

Table 1 presents a summary of the most popular offline evaluation metrics used for recommendation systems. Offline evaluation can be thought of as either a classification or a regression (prediction) problem, depending on the type of target variable we have (Herlocker, Konstan, Terveen, & Riedl, 2004). In the former, we are usually working with binary item consumption, and in the latter, we are usually working with explicit or implicit ratings.

Regression-type metrics for recommender system evaluation include the mean squared error (MSE), root mean squared error (RMSE), and normalized root mean squared error (NRMSE) (Katsov, 2018). For classification type models, we can employ multi-class/multilabel versions of traditional classification metrics such as F1-Score and AUC, which can usually be done by macro (across classes) or micro (across cases) averaging (Tsoumakas & Vlahavas, 2007).

Consider a binary evaluation measure $M(tp, tn, fp, fn)$ that is calculated based on the number of true positives (tp), true negatives (tn), false positives (fp) and false negatives (fn). Let tp_ρ, fp_ρ, tn_ρ and fn_ρ be the number of true positives, false positives, true negatives, and false negatives, respectively, after binary evaluation for a label ρ. The macro-averaged and micro-averaged versions of M, are calculated as follows, where L is the set of labels (items) (Tsoumakas & Vlahavas, 2007):

$$M_{macro} = \frac{1}{|L|}\sum_{\rho=1}^{|L|} M(tp_\rho, fp_\rho, tn_\rho, fn_\rho) \tag{1}$$

$$M_{micro} = M\left(\sum_{\rho=1}^{|L|} tp_\rho, \sum_{\rho=1}^{|L|} tn_\rho, \sum_{\rho=1}^{|L|} fp_\rho, \sum_{\rho=1}^{|L|} fn_\rho\right) \tag{2}$$

2.2. Online evaluation metrics

Online evaluation metrics can overcome these issues by testing the recommender system in a production environment. In this context, we can apply several specific recommender system metrics, usually derived from evaluation metrics applied in online advertising, e-commerce, and other transactional settings (Beel & Langer, 2015; Gomez-Uribe & Hunt, 2016a). Some of the most popular online evaluation metrics are the click-through rate (CTR) [7], recommendation take rate (Gomez-Uribe & Hunt, 2016a), and profit (Katsov, 2018).

Table 1
Most popular offline evaluation metrics.

Metric	Reference
Mean Squared Error (MSE) Root Mean Squared Error (RMSE) Normalized Root Mean Squared Error (NRMSE) F1 Score AUC	Katsov (2018)
Gini Coefficient Kolmogorov-Smirnov (KS) statistic	Verbeke, Dejaeger, Martens, Hur, and Baesens (2012)
Hamming Loss Jaccard Index	Luaces, Díez, Barranquero, del Coz, and Bahamonde (2012)
Diversity Coverage Serendipity Novelty	Katsov (2018)
Timeliness	Zhang, Liu, and Zeng (2017)
Dynamism	Lu, Dumitrache, and Graus (2020)

2.3. User studies evaluation metrics

User studies, which employ survey-based methods, come from the usability and user experience research tradition. Several standardized psychometric constructs exist to evaluate recommender systems namely the ResQues framework (Pu & Chen, 2010) and the subjective user experience (Knijnenburg, Willemsen, Gantner, Soncu, & Newell, 2012).

3. Non-transactional settings

Most of the metrics we have seen are also valid in non-transactional settings, with the notable exception of the profit.

In non-transactional commercial environments (including email, news, and multimedia content recommendations, amongst others), profit is hard to directly associate with the consumption of a non-transactional item. For instance, in mobile app recommendations, most apps are free to download (94.24% in the Android platform, 88.18% in iOS) (Statista, 2018), which means that the profit can only be used in a small proportion of paid apps. For non-paid items, we need to consider a different approach to profit measurement based on the recommender system's impact on the customer lifetime value (CLV) (Iwata, Saito, & Yamada, 2008), a key concept in customer relationship management (CRM) (Blattberg, Kim, & Neslin, 2008).

The basic rationale for estimating a non-transactional recommendation's profit is that the recommender system ought to improve consumption frequency and reduce churn, thereby, impacting the CLV positively.

CLV can be calculated using several methods. Traditionally, CLV would be estimated using a recency, frequency and monetary (RFM) model to rank users (Gupta et al., 2006). Modern approaches extend the concept of RFM to obtain a financial estimation of the CLV for each customer. One approach to obtaining a global average of CLV is to calculate the average of all individual-level CLV values across a database, where the CLV for each customer is given by Blattberg et al. (2008):

$$CLV = \sum_{t=1}^{\infty} \frac{\left(\tilde{D}_t - C_t\right) S_t^{t-1}}{(1+\delta)^{t-1}} \tag{3}$$

where δ is the discount rate per time unit t, $\tilde{D}_t$ is revenue the user generates in moment t, C_t is the cost of serving user in moment t, and S_t is the probability of the customer not churning before moment t. The RFM model is incorporated into CLV through the S_t ("recency" and "frequency") and $\tilde{D}_t$ ("monetary"). Several methods to obtain $\tilde{D}_t$ have been proposed (Blattberg et al., 2008).

To model $S_t = p(alive)$ we need a consumer behavior model such as the beta binomial/NBD (BB-NBD) (Jeuland, Bass, & Wright, 1980), NBD-Dirichlet (Goodhardt, Ehrenberg, & Chatfield, 1984), Pareto/NBD (P-NBD) (Schmittlein, Morrison, & Colombo, 1987), beta-geometric/beta-binomial (BG-BB) (Fader, Hardie, & Berger, 2004), and beta-geometric/NBD (BG-NDB) (Fader, Hardie, & Lee, 2005).

We will focus our attention on the NBD-Dirichlet model (Goodhardt et al., 1984) given its simple data requirements.

NBD-Dirichlet results from the combination of two distributions: the negative binomial distribution (NBD) and the Dirichlet multinomial distribution (DMD) (Dawes, Meyer-Waarden, & Driesener, 2015). The NBD part describes individuals' category-buying behavior in a market, and the DMD part models the probability of each individual in the market purchasing a specific brand Goodhardt et al. (1984). The resulting model is therefore given by Goodhardt et al. (1984):

$$p\left(r_j|n\right) = \frac{\binom{n}{r_j} B\left(\alpha_j + r_j, S - \alpha_j + n - r_j\right)}{B\left(\alpha_j, S - \alpha_j\right)} \tag{4}$$

where $p\left(r_j|n\right)$ can be interpreted in the context of online retail as the probability of using r_j times the retailer j among n uses of the retailer category, α_j is the use propensity of the retailer j, and S is the diversity of usage behavior in the category ($S = \sum_j \alpha_j$) (Bound, 2009). We can assume that $MS_j = \alpha_j / S$ where MS_j is the market share of the retailer j (Wright, Sharp, & Sharp, 2002). B is the Beta function such that:

$$B(p,q) = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)} \tag{5}$$

Γ is the gamma function such that:

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t} dt, x > 0 \tag{6}$$

Accurate estimates for all α_j can be obtained by fitting the DMD for all retailers J in the market using panel data such that Wrigley and Dunn (1985):

$$p\left(r_1, r_2, \ldots, r_j|n\right) = \binom{n}{r_1, r_2, \ldots, r_j} \frac{\Gamma(S)}{\Gamma(S+n)} \prod_{j=1}^{J} \left[\frac{\Gamma\left(\alpha_j + r_j\right)}{\Gamma\left(\alpha_j\right)}\right] \tag{7}$$

Model estimation can be done using the original "mean and zero" method (Goodhardt et al., 1984) proposed or maximum likelihood (Wrigley & Dunn, 1985). Intelligence providers such as App Annie and 42Matters provide app usage intelligence data for the mobile industry, which can be used to estimate these parameters for all platforms. An Excel workbook is available to facilitate the model estimation using maximum likelihood (Rungie, 2003). An R package is also available (Cheng et al., 2016), which can be used to obtain estimates of S by having as input the platform category penetration (category users in the population), platform penetration (users of the specific platform in the population), category use frequency and the various platforms' market shares. We can then obtain the platform use propensity through its market share ($MS_j = \alpha_j / S$). The required input information can be found in several industry publications and market intelligence providers such

as Statista and GSMA Mobile Economy. Under this set of assumptions, $p(alive)$ is $p\left(r_j > 0|n\right)$ where n is the average usage frequency of the retailer category. NBD-Dirichlet is a realistic assumption because online consumer behavior in the context of mobile app stores has been shown to follow the double jeopardy law and natural monopoly effects the NBD-Dirichlet model describes (Zhong & Michahelles, 2013).

3.1. Action value estimator

We propose a novel offline profit-based metric based on the concept of the estimated average action value ($\hat{V}_{action}$), derived from CLV metrics and the NBD-Dirichlet.

Let us begin by considering the main assumptions behind the NBD-Dirichlet model (Goodhardt et al., 1984), which are grounded in the empirical generalizations that have been observed over several decades across markets and geographies (Graham, Bennett, Franke, Henfrey, & Nagy-Hamada, 2017). In particular, these patterns have also been observed in several types of online platforms, such as the mobile app economy (Zhong & Michahelles, 2013) and amongst online retailers (Huang, 2011). Therefore, we believe these to be a set of realistic assumptions for the online context.

1. The probability of a user performing k independent actions in a period follows a Poisson distribution.

$$p(s_{ijt} = k) \approx \frac{e^{-\mu_i}\mu_i^k}{k!} \tag{8}$$

where $s_{ij} \in N$ is the number of actions a user i has done in period t in an online platform j and μ_i is the average action frequency per period of user i.

2. The action frequencies of individual user i action varies according to a gamma distribution.

$$\tilde{s}_i \approx \frac{e^{-\mu_i \frac{K}{M}}\mu_i^{K-1}}{\Gamma(K)(\frac{M}{K})^K} \tag{9}$$

where M is the platform category average use rate and K is its variance.

3. User's i probability of choice c_{ij} of online platform j over k successive choices follows a multinomial distribution.

$$c_{ij} \approx \frac{k!}{\prod_{t=1}^{T} S_{ijt}} \prod_{g \in Platforms} \alpha_g \tag{10}$$

where α_g is the average usage frequency of platform g.

4. Individual user i usage frequency of online platform g varies according to a Dirichlet distribution.

$$\theta_{ig} \approx \frac{\prod_{g \in Platforms}\left(\sum_{t=1}^{T} s_{igt}\right)^{\alpha_g - 1}\Gamma\left(\sum_{g \in Platforms}\alpha_g\right)}{\prod_{g \in Platforms}\Gamma\left(\alpha_g\right)} \tag{11}$$

5. The online platform choice probabilities and the average action frequencies of the various users are distributed independently.

Therefore, the probability $p(r_j|n)$ of a user i doing r_j actions in period t on online platform j amongst n_i actions done across all competing platforms is given as:

$$p(r_j|k) \approx \frac{\binom{k}{r_j} B(\alpha_j + r_j, S - \alpha_j + k - r_j)}{B(\alpha_j, S - \alpha_j)} \tag{12}$$

where S is the diversity of usage behavior in the category ($S = \sum_j \alpha_j$) (Bound, 2009). We can assume that $MarketShare_j = \alpha_j / S$ (Wright et al., 2002).

From this, we can obtain a conditional probability distribution for $p(alive|k)$:

$$p(alive|k) = 1 - p(r_j = 0|k) \tag{13}$$

Additionally, let CLV_{i_T} be the customer lifetime value of user i according to the expression Blattberg et al. (2008) proposed:

$$CLV_{i_T} = \sum_{t=1}^{T} \frac{(\tilde{D}_{it} - C_{it})S_t^{t-1}}{(1+\delta)^{t-1}} \tag{14}$$

where $\tilde{D}_{it}$ is the revenue user i generates in period t, C_{it} is the cost of serving the user i in moment t, and δ is the discount rate, and T is a finite time horizon. By replacing $S_t = p(alive)$ with our probability density function for $p(alive)$, we get our V_{action} estimator:

$$\hat{V}_{Action} = \sum_{t=1}^{T} \frac{(\tilde{D}_{it} - C_{it})}{(1+\delta)^{t-1}} \times \left(1 - \frac{B(\alpha_j, S - \alpha_j + k)}{B(\alpha_j, S - \alpha_j)}\right)^{t-1} \tag{15}$$

We should note that the NBD-Dirichlet model was designed for the fast moving consumer goods (FMCG) sector in which the buying behavior tends to occur with a weekly periodicity, which makes assumption 1. appear more plausible (Lees, 2009). However, that is not the case in many other sectors, especially in non-transactional digital environments, where the behavior might be more clumpy (Zhang, Bradlow, & Small, 2015). Clumpiness is observed in several types of online platforms for instance, most notably in streaming platforms such as Netflix, where users "binge" several pieces of content in a short period of time and then go several weeks without using the platform (Lu, Karmarkar, & Venkatraman, 2019). It is also plausible that a similar behavior might occur for some users in mobile app platforms such as the one we are presenting in this application, we believe it is less common. For instance, we know from publicly available information that the average user in the US downloads only three apps per month (App Annie, 2018). Another relevant aspect is the period length considered. We know that increasing or decreasing its length might impact the observed clumpiness effect (Zhang et al., 2015). That said, the existing empirical applications of the NBD-Dirichlet model in the mobile app industry seem to confirm that the Poisson assumption still holds for those contexts (Stocchi, Guerini, & Michaelidou, 2017; Zhong & Michahelles, 2013).

4. Recommender system profit

We can employ the action value estimator defined in the previous section to obtain a profit metric for recommender systems. In this application context (mobile app stores), we can use the app *download* as the reference action, which will means we can define a download value, $\hat{V}_{download}$, from the general action value estimator expression proposed.

We will assume that any true positive results in an increment of $+\hat{V}_{download}$ (revenue) and any false positive is counted as a decrement of $-\hat{V}_{download}$ (opportunity cost). However, we must consider additional cost variables that were added to compare each models' efficiency considering the model training, validation, testing and deployment.

The final profit value for a recommender system model can then be estimated using the following expression:

$$Revenue_{model} = \hat{V}_{download}(U\tau_{model} - U\psi_{model}) \tag{16}$$

$$Profit_{model} = Revenue_{model} - tUCost_{deployment} - Cost_{trn;val;tst} \tag{17}$$

where U is the size of the test set, τ_{model} is the true positive rate, ψ_{model} is the false positive rate, t is the deployed model execution time, $Cost_{deployment}$ is the cost of the model deployment (per unit of time), and $Cost_{trn;val;tst}$ is the absolute cost of model training, validation and testing.

The values for $Cost_{deployment}$ and $Cost_{trn;val;tst}$ come from the cloud platforms hourly costs, which we will be discussed in the following sections. We will estimate the deployed model execution time (t) using a separate experiment conducted in a Spark environment. In this case, we are interested in the expected execution time for the entire Spark job associated with the model deployment. Remember that Spark jobs are made of multiple stages, and that each stage contains several tasks

running in parallel in a distributed manner (Wang & Khan, 2015). The TensorFlow model to be deployed is encapsulated in an object that will require only a single stage, with several tasks. The total job execution time is therefore given by the following expressions (Wang & Khan, 2015):

$$P = \sum_{i=1}^{H} CoreNum_i \tag{18}$$

$$JobTime = JobStartup + StageTime + JobCleanup \tag{19}$$

$$StageTime = StageStartup + max_{v=1}^{P} \sum_{i=1}^{R_v} TaskTime_{v,i} + StageCleanup \tag{20}$$

where $CoreNum$ is the number of CPU cores of working node i, and H is the number of working nodes in the cluster, R_v is the number of sequential tasks executed on CPU core v and P is the total number of CPU cores in the Spark cluster. The number of sequential tasks R_v under these conditions will be the number of user batches (RDD partitions) included in the testing experiment divided by the number of CPU cores. We can separate the total job time into two components as such:

$$Z = JobStartup + StageStartup + StageCleanup + JobCleanup \tag{21}$$

$$V = max_{v=1}^{P} \sum_{i=1}^{R_v} TaskTime_{v,i} \tag{22}$$

$$JobTime = Z + V \tag{23}$$

Because this experiment is to be executed in a single machine cluster, we will simulate the typical industrial cluster configurations by estimating the V and Z components using Ψ sequential trials (the number of trials meant to simulate the number of nodes in the cluster). To simplify the calculations, we will employ a single processing CPU core. We can define two finite sets, $V = \{v_1, \ldots, v_\Psi\}$ and $Z = \{z_1, \ldots, z_\Psi\}$, which represent the ψ observations of Z and V. Each element of V represents the sum of all task durations in the trial. From these sets, we can define two estimators for the actual values. For Z we are looking for the expected value $\mu(Z)$. For V we are looking for the maximum expected value, $\mu_*(V)$, which can be estimated using the maximum estimator (ME) (van Hasselt, 2013). Therefore, we have:

$$V \approx \hat{V} \equiv \hat{\mu}_*(V) \equiv max(V) \tag{24}$$

$$Z \approx \hat{Z} \equiv \hat{\mu}(Z) \equiv \frac{\sum \mathcal{Z}}{|\mathcal{Z}|} \tag{25}$$

The final $\hat{t}$ will then be given as:

$$\hat{t} = max(V) \frac{\sum \mathcal{Z}}{|\mathcal{Z}|} \tag{26}$$

To simulate the typical big data setups, we will assume $\Psi = 12$. Wang & Khan 2015 used this cluster size in their experiment, and it was consistent with the reported industry best practices (Fujitsu, 2017). Even though much larger cluster sizes may occur for specific tasks (Apache Foundation, 2018), they do not appear to be common. We extracted the runtimes from the Spark Web UI, which displays runtimes for tasks, stages, and jobs. We will assume $\xi = 2.958$ USD/h since it is the current pricing of a general purpose m5 12 node cluster on the most popular infrastructure provider, Amazon Web Services (AWS), which includes the EC2 and EMR costs (Amazon Web Services, 2018).

5. Research design

An empirical study was conducted to illustrate how our proposed metric can be used to evaluate recommender systems in big data environments. This section describes the data pre-processing, experimental settings, and the obtained results.

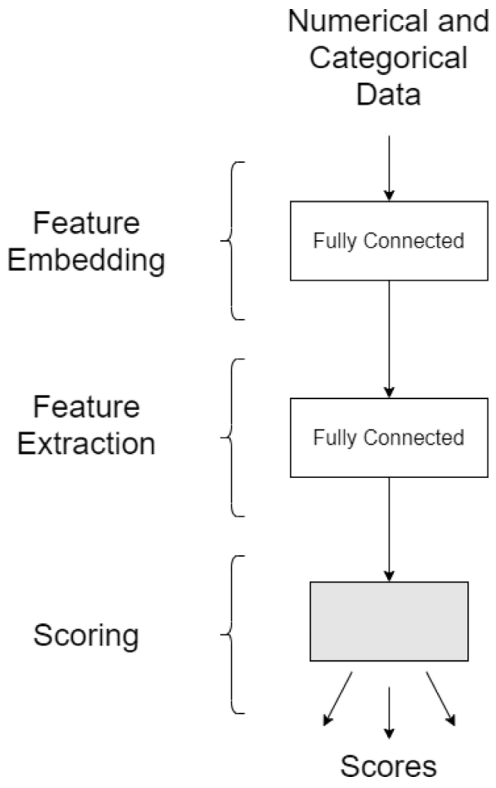


Fig. 1. Deep-Learning Architecture Overview.

5.1. Experimental design

An empirical study was conducted on a Portuguese Android app store, Aptoide. This platform is presented as a "social app store", where some common online social network features exist. Users can create their profiles and follow each other, and they have access to a microblogging feature (through the "apps timeline" feature, which was still active at the time of the dataset extraction), user-generated comments and app reviews. Additionally, users can download and share apps with their followers. It also has a search engine to find Android apps. Several models are compared using the proposed metric. We will be testing a generic architecture for mobile app recommendations with three basic layers: feature embedding, feature extraction, and scoring. We defined the different deep learning models to facilitate comparisons across these three levels (Fig. 1). These different fully connected deep architectures are designed to score mobile apps for users using aggregated features at the user level, similar to other previously architectures (Kiran, Kumar, & Bhasker, 2020) proposed.

The specific goals of this experiment are threefold:

1. To show that the proposed performance metric (profit) behaves in a manner generally consistent with existing metrics.

2. To show that the proposed metric is useful to distinguish between similarly performing models using an objective economic criteria.

3. To show that the proposed metric is stable and robust to changes in its parameters.

Table 2 presents a description of the four models. The task in our experiment consists of scoring a batch of 10 apps for each user as a multilabel classification problem. We recommend an app if the score is over a threshold (commonly 0.5). If the user has previously acquired the app, we consider it a true positive. If not, we consider it a false positive. We apply the same logic to negatives. The results of each model on this task will answer our research questions. Figs. 2, 3, and 4 present the overall schema used for models 1,2,3, and 4. To answer research question 1, we will compare the performance of models 1 and

Table 2
Brief description of the models used.

Model 1	Three-layer network with a fully connected (FC) feature-embedding layer (numeric features), feature-extraction layer, SoftMax output, and no pre-training.
Model 2	Four-layer network with one layer of FC-feature embedding (numeric and categorical features) and two feature-extraction layers, SoftMax output, and no pre-training.
Model 3	Four-layer network with one FC-feature-embedding (numeric and categorical features) and two feature-extraction layers, SoftMax output, and embedding pre-training.
Model 4	Four-layer network with one FC-feature-embedding (numeric and categorical features) and two feature-extraction layers, kernel SoftMax output, and embedding pre-training.

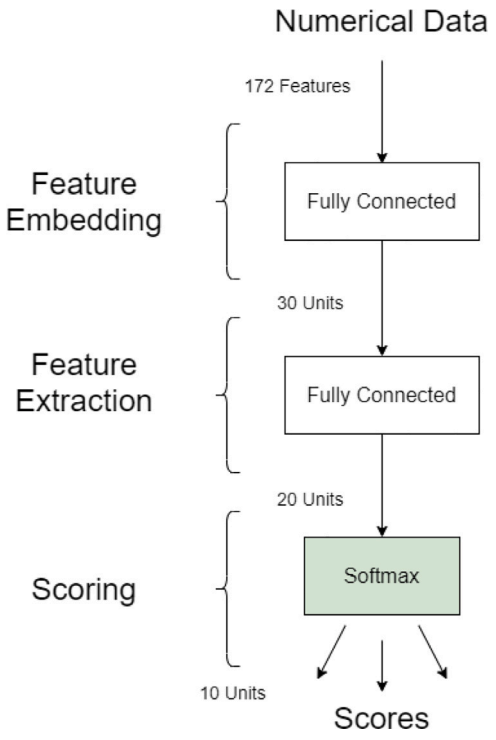


Fig. 2. Architecture of model 1.

2. To answer research question 2, we will compare the performance of models 2 and 3. To answer research question 3, we will compare the performance of models 3 and 4. We will present the profit values normalized across models to protect internal financial information. A profit of zero (0%) implies the worst model, while a profit of 1 (or 100%) implies the best model.

The first three layers are fully connected components with ReLU activation functions. The embedding layer is pre-trained using a combined Word2Vec and Autoencoder procedure from numerical raw features and one-hot encoded categorical variables. The resulting embeddings are fed into the following layer. The feature extraction layers take the embedded features for each user in each moment and performs a dimensionality reduction. These features will be used as input for the final scoring layer. Two variants of the architecture can be constructed using different scoring methods. The standard approach is to employ SoftMax. As a novel approach to recommender system design, we propose an alternative multi-class classification method based on Kernel methods. For this purpose, we will use TensorFlow's implementation of Kernel methods, which is based on random Fourier features (RFF) (Rahimi & Recht, 2007). By combining RFF with a SoftMax output node, we can implement a deep version of a multi-class kernel

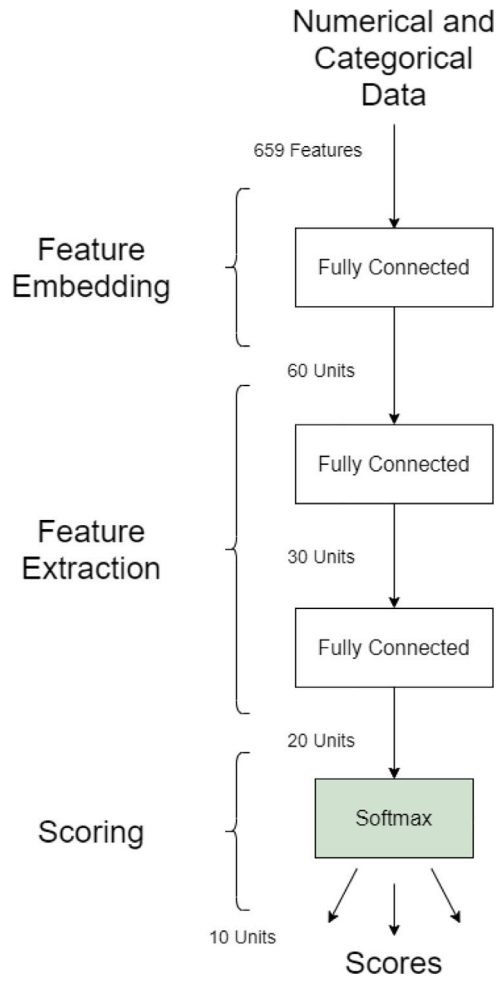


Fig. 3. Architecture of model 2 and 3.

Table 3
Brief description of the models used.

Geographic	Demographic	Technographic	Behavioral	Social
Latitude Longitude	Language	Android Vers. Device Modl.	# Downloads # Searches # Clicks App Downloads Search Queries Clicked Items	Social Network Topology

logistic regression (M-KLR) model (Karsmakers, Pelckmans, & Suykens, 2007). TensorFlow does not currently offer a differentiable SVM or multi-class SVM implementation. However, kernel logistic regression (KLR) has been empirically and analytically demonstrated as having similar performance and behavior similar to SVM's (Karsmakers et al., 2007), the main difference being the fact that it requires the entire dataset as opposed to only using support vectors to build a decision margin (Zhu & Hastie, 2005). As such, it is expected that M-KLR should behave similarly to multi-class SVM approaches.

6. Data

We extracted the dataset we employed to train and test the various models from Aptoide's big data lake running on Amazon Web Services (AWS). Table 3 lists the raw features available in the dataset, which were subjected to a pre-processing stage.

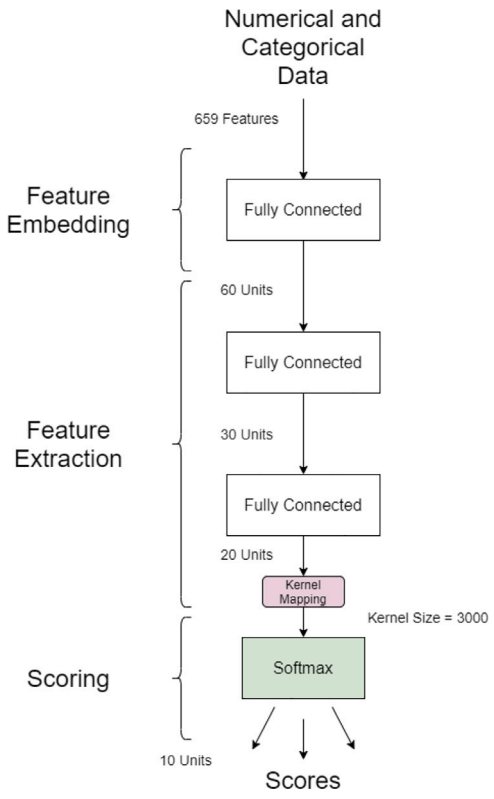


Fig. 4. Architecture of model 4.

7. Empirical analysis

We divided the processed dataset into three parts making up the training, validation, and test sets, roughly corresponding to 86%, 2% and 12% of the overall data (74,209 users). We conducted the model training, validation, and testing on an Nvidia Tesla K80 GPU instance executing a TensorFlow environment using the Floyd Hub service (https://www.floydhub.com/).

7.1. Model training and validation

The training of models 3 and 4 involved a pre-training of the feature-embedding architectural component using a Word2Vec (for the unstructured data) and Autoencoder implementation in TensorFlow. We trained both models over ten epochs with gradient descent. We concatenated the resulting embedding parameters (we randomly initialized the cross-loadings with lower values close to zero) and used them as the initialization tensor for the first layer of the final architecture. We followed this pre-training with a global training stage. We divided the training set into 32 batches of 2.000 users (64.000 overall). We conducted the training procedure under the classic early stopping method (Prechelt, 2012), which is dependent on the loss on the entire validation set computed at each epoch (if the validation loss increases in this epoch, the training stops, and the previous weights are retained). We used the Adam optimizer with the standard parameters (Kingma & Ba, 2014) to perform the network training backpropagation. Fig. 5 presents the loss evolution for each model for the training and validation data. We are reporting the absolute loss for every model and stage, which explains the fact that the training loss is consistently higher. It merely reflects the fact that the training set size is larger than the validation set. We have also noted that the loss for model 2 is non-smooth. We believe it is related with the fact that this model was not subjected to a pre-training stage, while at the same time being fed several hundreds of categorical (one-hot-encoded) features. As evidence of this we can see that model 1 (not subjected to a pre-training stage) also exhibits some non-smoothness, albeit less, which can be explained by the fact that it was fed only numerical features.

Table 4
Computational statistics of model training, validation and testing.

	Model 1	Model 2	Model 3	Model 4
Max CPU utilization	56%	72%	74%	**55%**
Execution time (min:sec)	03:34	**03:22**	05:35	04:38
Total cost of training, validation and testing (USD)	$0.071	$0.067	$0.111	$0.093

Table 5
Computational statistics of model deployment.

	Model 1	Model 2	Model 3	Model 4
Estimated serving latency (ms)	**7.56**	8.45	15.11	15.34
Cost per user (USD)	**3.10E−04**	3.47E−04	6.21E−04	6.30E−04
Total cost of deployment (USD)	$0.70	$0.78	$1.40	$1.42

7.2. Model testing

We conducted the model testing in two parts. The first part was meant to test the model accuracy using standard metrics on the testing set on the Nvidia Tesla K80 GPU instance where the training and validation were done. We used the average pricing per hour of the infrastructure provider (Floyd hub) to compute the total costs of training, validation, and testing (1.22 USD/h) (Table 4).

We then allocated a portion of the testing set (2250 users) to a secondary test to obtain the Spark execution time component (Table 5).

We measured the Spark estimated serving latency separately on an AMD quad-core Processor A4-5000 (1.5 GHz) CPU machine executing a Spark server on an Ubuntu VM through a Windows 10 host. We used a single CPU core of this machine to execute the experiment, and we used the obtained execution time for each model to estimate the information technology (IT) infrastructure cost component. A Spark script was executed directly on the PySpark command-line tool, which imports the TensorFlow graph from the local disk before applying it to each row of an RDD, according to the method Databricks proposed (Hunter, 2016).

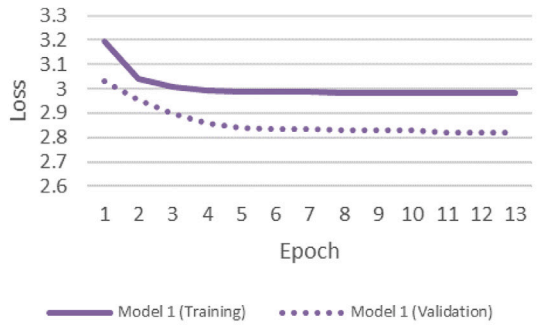
The remaining $\hat{V}_{download}$ estimator parameters $(\tilde{D}_{it}, C_{it}, \delta, h)$ were obtained from internal company data, and the parameters $\alpha_j = 0.01$, $k = 1$, and $S = 0.08$ were obtained from public sources (App Annie, 2018).

Fig. 6 presents the values from both tests we combined to compute the final profit value for each model. We also compared the Hamming loss, Jaccard index, AUC (Macro), F1 score (Macro and Micro), and precision and recall across models.

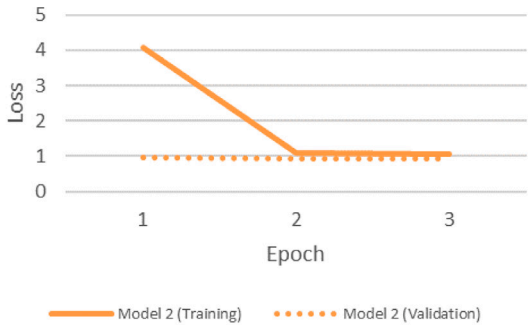
Table 6 presents the results of the testing for each model (bold indicates top performer).

The results show that model 1 is the lowest performer in all metrics except recall (in which it significantly outperforms model 2). Except for recall and Hamming loss, models 1 and 2 behave similarly, but model 2 beats model 1 overall. Models 3 and 4 outperform the other two models significantly across all metrics. The difference between models 3's and model 4's performance is less clear. These two models seem to have a small difference magnitude overall. Still, model 3 outperforms model 4 in most metrics except F1 score (Macro), recall, true positive rate and false negative rate.

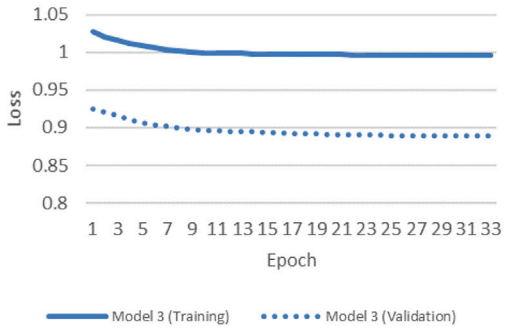
Here we can see the value of the proposed profit metric, since it helps us to have an objective business perspective that solves the apparent tie between model 3 and 4. While the generic machine learning/data science metrics give several different (technical) perspectives,



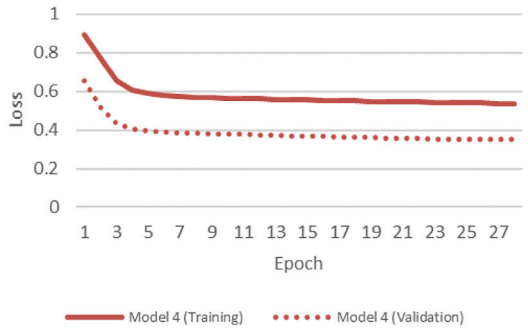
(a) Training and Validation Loss (Model 1).



(b) Training and Validation Loss (Model 2).



(c) Training and Validation Loss (Model 3).



(d) Training and Validation Loss (Model 4).

Fig. 5. Training and validation loss for all models.

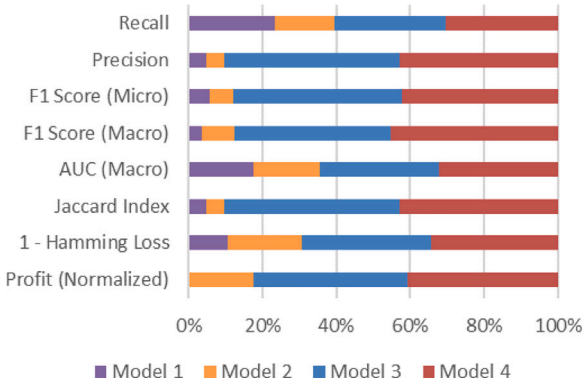


Fig. 6. Experimental results of model testing.

Table 6
Experimental results of model testing (in detail).

	Model 1	Model 2	Model 3	Model 4
Hamming loss	0.713	0.462	**0.065**	0.076
Jaccard index	0.031	0.033	**0.314**	0.284
AUC (Macro)	0.523	0.540	**0.973**	0.964
F1 score (Macro)	0.050	0.123	0.585	**0.626**
F1 score (Micro)	0.061	0.065	**0.478**	0.442
Precision	0.032	0.035	**0.320**	0.288
Recall	0.731	0.507	0.944	**0.951**
True positive rate	2.30%	1.60%	2.98%	**3.00%**
True negative rate	26.36%	52.24%	**90.54%**	89.45%
False positive rate	70.49%	44.61%	**6.31%**	7.40%
False negative rate	0.85%	1.56%	0.18%	**0.16%**
Profit (normalized)	0%	39%	**100%**	98%

the most relevant for a given firm will always be the business criteria. Observe for instance the F1 Score, which results in a different conclusion when Macro averaged vs when Micro averaged. From a purely technical point of view it will be hard to justify why one perspective is better than the other. Another example would be the AUC metric, which is extremely close between these two models. Given the absence of other more subjective criteria such as the ones presented in Table 1 (such as Diversity, Coverage, Serendipity, Novelty, Timeliness and Dynamism) we could have some doubts on what a slightly higher AUC metric might entail for the end user and for the business. The profit criteria provides a financial justification that is easy to understand.

By looking at Table 7, we can see that our proposed profit metric is highly correlated with 1-Hamming and the true negative rate (TNR), which are also highly correlated between them. This finding reflects the fact that the 1-Hamming loss is the proportion of correct recommendations to the total number of recommendations, which, for a highly unbalanced and sparse dataset such as this one, will mostly match the TNR. That said the correlation between profit and TNR/1-Hamming might be spurious, given that TNR is not used to compute the profit. This might explain why we have a lower (but still strong) correlation with AUC, which is widely considered to be the best metric to evaluate classifiers (Vanderlooy & Hüllermeier, 2008). The proposed metric might be useful as a way of interpreting the AUC financially.

7.3. Sensitivity analysis

The profit estimate of each model relies on the $\hat{V}_{Action}$ estimator which depends on the hyperparameters $\theta \in (\tilde{D}_{it}, C_{it}, \delta, \alpha_j, S, k)$. In this section we will analyze the impact of small changes of these parameters on the model profitability estimates. We will obtain a set of confidence intervals for the model profitability using a Monte Carlo simulation approach with 10,000 simulations. For each simulation we have a

Table 7
Correlation between performance metrics.

	Profit	1 - Hamming	Jaccard	AUC	F1 (Macro)	F1 (Micro)	Precision	Recall	TPR	TNR	FPR	FNR
Profit	1.0000	0.9994	0.9343	0.9434	0.9621	0.9356	0.9348	0.6855	0.6855	0.9998	−0.9998	−0.6855
1 - Hamming		1.0000	0.9454	0.9541	0.9707	0.9468	0.9458	0.7098	0.7098	0.9999	−0.9999	−0.7098
Jaccard			1.0000	0.9975	0.9864	0.9999	1.0000	0.8941	0.8941	0.9405	−0.9405	−0.8941
AUC				1.0000	0.9950	0.9985	0.9974	0.8881	0.8881	0.9495	−0.9495	−0.8881
F1 (Macro)					1.0000	0.9886	0.9861	0.8525	0.8525	0.9671	−0.9671	−0.8525
F1 (Micro)						1.0000	0.9998	0.8947	0.8947	0.9419	−0.9419	−0.8947
Precision							1.0000	0.8930	0.8930	0.9409	−0.9409	−0.8930
Recall								1.0000	1.0000	0.6992	−0.6992	−1.0000
TPR									1.0000	0.6992	−0.6992	−1.0000
TNR										1.0000	−1.0000	−0.6992
FPR											1.0000	0.6992
FNR												1.0000

Table 8
Confidence Intervals for the Profitability of each model.

	Model 1	Model 2	Model 3	Model 4
Upper (95% CI)	0.6%	42.5%	100.0%	98.5%
Profit (Point Est. w/ observed θ)	0.0%	42.1%	100.0%	97.9%
Lower (95% CI)	0.0%	41.8%	99.4%	97.2%

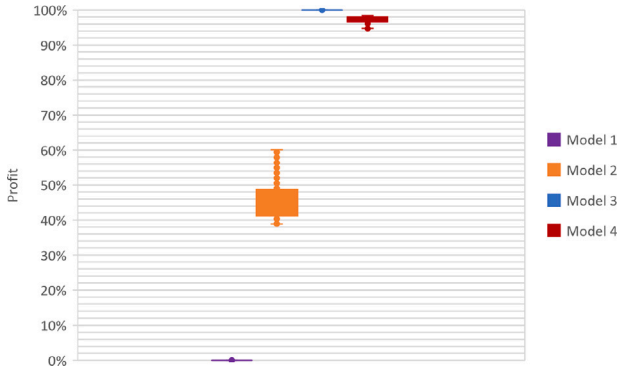


Fig. 7. Boxplot of profit simulation results.

perturbation term $\varsigma_m \sim Beta(\alpha = 2.8, \beta = 3.2)$.[1] These values will be used as the relative standard deviation of a normally distributed random variable $W_m \sim N(0, \theta \times \varsigma_m)$. This allows us to obtain a simulated parameter $\theta *_m = \theta + W_m$, and an associated $\hat{V}_{Action_m}$ for each Monte Carlo simulation.

From these values we computed the model profitability using Eq. (20) as before, and we constructed the confidence intervals by approximation to the Normal distribution.

Table 8 along with Fig. 7 appears to confirm the stability of the model profitability estimates. Additionally we can say that, under the assumptions of the Monte Carlo simulation, the profitability differences between models are statistically significant for a confidence level of 95%.

7.4. Benchmarking

In addition to the private dataset, we have also conducted an application of this novel metric to three existing public datasets in the movie and games domains, along with their current state of the art (SOTA) models: Movielens (SOTA: Glocal K Han, Lim, Long, Burgstaller, & Poon, 2021), Netflix Prize (SOTA: H+Vamped Gate Kim & Suh, 2019), and Steam (SOTA: SASRec Kang & McAuley, 2018). Since the metric requires business context inputs, we have also considered two different markets: Streaming video on demand (SVoD) and transaction gaming on demand (TGoD). This allows us to test the metric in both transactional and non-transactional settings, across two different recommendation domains (movies and games). Within the SVoD category we considered following platforms: Netflix, HBO Max, Disney+, and Hulu, while in the TGoD category we considered Steam and Epic Games. By repeating the experiments of the SOTA models, and by applying the different business context inputs, we have obtained the following benchmark table (Table 9).

To obtain these values we used the open source code for the aforementioned SOTA experiments (Han et al., 2021; Kang & McAuley, 2018; Kim & Suh, 2019) and the publicly available datasets used in those works (Movielens, Netflix Prize and Steam), we reproduced the experiments in a local Intel Core i7 machine running Windows 10 to obtain the values for true positives and false positives, required to compute the revenue component of our metric.

The platform business context information was obtained by combining several different public sources with data from late 2021 and early 2022 (Carson & Brady, 2021; Obedkov, 2022; Park Associates and Symphony Media A.I., 2021; Rayburn, 2022; Statista, 2022b; Wise, 2022a, 2022b). The platform usage frequency (α_j) was estimated using the observed (Carson & Brady, 2021) and theoretical churn rates from the Dirichlet expected churn Eq. (13) by minimizing the sum of the squared errors. The value of $\tilde{D}_{it} - C_{it}$ was approximated using the publicly available ARPU estimates for each platform (Rayburn, 2022). The final action values were then obtained by applying Eq. (15).

The previously introduced sensitivity analysis methodology was applied to these experiments to obtain simulated variance estimates for the different SOTA model/business/dataset combination. A meta-analysis (Table 9) was applied to the resulting data to obtain pooled estimates of the revenue (16) at 10 recommendations (Rev@10) for the different platforms (Netflix, HBO Max, Disney+, Hulu, Steam and Epic Games) (Table 10 and Fig. 8).

We have compared the resulting estimates with public earning report information for the year 2021 for Netflix (Netflix, 2022), as well as public usage and revenue statistics for Netflix, Hulu, and HBO Max in the US (Curry, 2023a, 2023b; Georgiev, 2023; Statista, 2023a, 2023b; Susic, 2023), as well as general US TV viewing statistics (Statista, 2022a), which point to an average 3 h of TV usage, of which 35% are on streaming platforms (Fuhrer, 2022). We have also considered the results by Gomez-Uribe and Hunt (2016b) which point to approximately 80% of hours streamed on Netflix coming from recommendations, with only 20% coming from search. We assumed that each unit of content (either movie or series episode) has an average duration of 42 min, taken from public information about the Netflix catalog (Moore, 2020; Pressman, 2015). With these figures, we were able to obtain an estimated "average revenue per recommendation", assuming that the 80% statistic holds not just for Netflix but also for Hulu and HBO Max. The results, along with the comparison with the estimates using our methodology are presented in Fig. 9. Disney+ was not included in this analysis due to insufficient availability of public information.

[1] The α and β parameters of the Beta distribution were obtained using the well known PERT reparameterization formulas (Farnum & Stanton, 1987). We have subjectively assumed $a = 10\%$ (small deviation), $b = 50\%$ (expected deviation) and $c = 100\%$ (large deviation).

Table 9
Benchmark results.

Model	Platform	Market	Action Value	Domain	Dataset	SotA Model	TP@10	FP@10	Rev@10	Rev-Opp.Cost@10	Sim. Variance	Weight
1	Netflix	SVoD	$114.17	Movies	Movielens	Glocal K	501	7728	$6.95	−$6.10	6.34E−02	16
2	HBO Max	SVoD	$86.06	Movies	Movielens	Glocal K	501	7728	$5.24	−$4.60	1.43E−03	697
3	Disney +	SVoD	$32.92	Movies	Movielens	Glocal K	501	7728	$2.00	−$1.76	1.41E−03	708
4	Hulu	SVoD	$97.78	Movies	Movielens	Glocal K	501	7728	$5.95	−$5.23	1.49E−03	673
5	Netflix	SVoD	$114.17	Movies	Netflix Prize	H + Vamped Gate	6897	392913	$1.97	−$1.90	5.48E−08	18,238,557
6	HBO Max	SVoD	$86.06	Movies	Netflix Prize	H + Vamped Gate	6897	392913	$1.48	−$1.43	8.19E−08	12,207,121
7	Disney +	SVoD	$32.92	Movies	Netflix Prize	H + Vamped Gate	6897	392913	$0.57	−$0.55	3.28E−09	305,315,544
8	Hulu	SVoD	$97.78	Movies	Netflix Prize	H + Vamped Gate	6897	392913	$1.69	−$1.63	4.82E−08	20,746,328
9	Steam	TGoD	$25.75	Games	Steam	SASRec	8705	90725	$2.25	−$1.86	7.35E−07	1,359,739
10	Epic Games	TGoD	$12.25	Games	Steam	SASRec	8705	90725	$1.07	−$0.88	9.72E−07	1,028,920

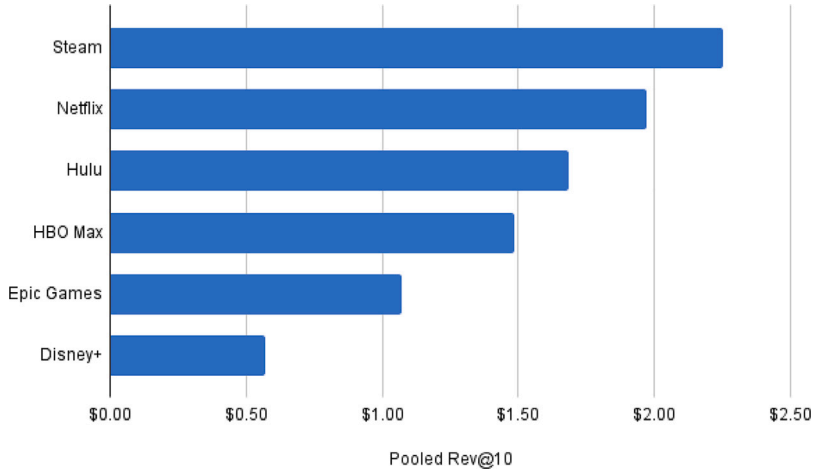


Fig. 8. Comparison of pooled Rev@10 estimates by platform.

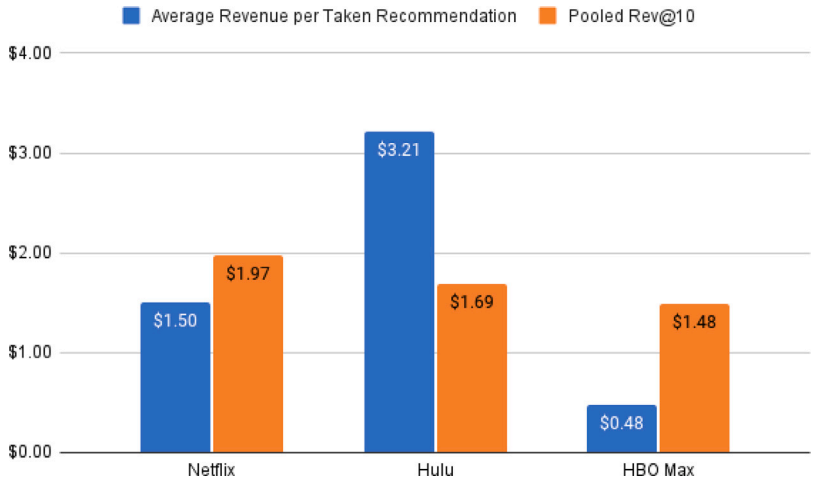


Fig. 9. Comparison of average revenue per taken recommendation and pooled Rev@10 estimates by platform.

Table 10
Platform pooled Rev@10 estimates.

Platform	Lower (95%CI)	Pooled mean	Upper (95%CI)	Pooled Stdev
Netflix	$1.97	$1.97	$1.97	$0.04
HBO Max	$1.48	$1.48	$1.48	$0.01
Disney +	$0.57	$0.57	$0.57	$0.01
Hulu	$1.69	$1.69	$1.69	$0.01
Steam	$2.25	$2.25	$2.25	$0.00
Epic games	$1.07	$1.07	$1.07	$0.00

The results appear to be close for Netflix, but some discrepancies are observed in the figures for Hulu and HBO Max. While the overall trend for Netflix and HBO Max is observed, Hulu appears as the platform with most profitability per recommendation. These discrepancies might be related with a number of factors:

1. The number of taken recommendations were estimated using the reported number of subscribers. The actual number of individual viewers is usually much higher. For instance, we are able to obtain a closer figure for Hulu (1.41$) by using the estimated number of viewers in 2021 (9.6 millions). However, the figures for Netflix and HBO Max do not improve by using the same approach.

2. It is entirely possible that the take rate for the recommender systems of Hulu and HBO Max are significantly below the 80% figure reported for Netflix. For instance, if we consider a scenario of having a take rate of 26% for HBO Max, the figures match the pooled Rev@10 estimates.

3. All estimates for the average revenue for taken recommendation as well as Rev@10 rely on public information, most of which is reported by third parties, and not by the platforms themselves. This means that we have now way of verifying the quality of the source information. It is plausible that some of these estimates have a large error, which would influence the reported estimates under both methods.

4. The different platforms have different revenue models. While we tried to isolate the effects of the subscription revenue (by removing the advertising revenue from the Hulu earnings) to have comparable revenue estimates, in the particular case of HBO Max, we still have transactional revenue, which we were not able to separate using the existing information.

8. Conclusion and future directions

With this study, we intended to demonstrate the need to use profitability metrics in recommender system evaluation.

We have introduced a novel recommender system evaluation framework, and we have tested it in a private dataset with four different deep-learning models in the Android apps domain, as well as in two different public datasets in the movies and games domains using SOTA models. The former experiment also included a model deployment experiment in a simulated Big Data environment, which, combined with internal business information from Aptoide allowed us to test the novel evaluation metric, while comparing it with other existing and widely used metrics. The latter experiment used public business information of large SVoD and TGoD platform, which allowed us to compare the evaluation metric with public earnings data.

In the first experiment, the profitability metric allowed us to identify the model that had the most significant business benefit for this specific firm. The standard offline recommender system evaluation metrics did not make entirely clear which model was the best. The profitability metric we have proposed might be of use not only for recommender system evaluation but might also be extended to many different contexts of multi-class prediction. The metric also provides a universal language that non-technical product of marketing managers can understand. Therefore, we believe this work also has implications for IT/IS project managers tasked with implementing recommender systems in the context of mobile app stores. The concept of average app download value might be helpful in the evaluation of the earned value of the modeling tasks within the project, according to a traditional earned value management (EVM) framework (Batselier & Vanhoucke, 2017) widely used in project management (Project Management Institute, 2017). From a broader information management perspective, this concept might also be useful in evaluating the recommender system's impact across the information technology, information system, business process, business benefit and business strategy domains (Bytheway, 2014), because it incorporates high-level market effects (consumer behavior and competitive environment) which can then be compared to the overall technological infrastructure costs, as in our profit measure.

This first experiment has several limitations. The main limitation has to do with the fact that we have only employed offline evaluation of the models due to the costs associated with deploying a recommender system, which made online testing infeasible. Such methods, however, would have allowed us to measure other properties of the recommender system other than its accuracy and economic efficiencies, such as novelty and serendipity (the "usefulness" metrics (Lu et al., 2020)), and to compare them with our estimated profit metric. The authors intend to address this limitation in a future study, that will compare the Profit obtained in an offline setting with the actual Profit generated in an online context. The second limitation is the task used to test the models. We considered a batch of 10 apps that were being scored for all users. In other similar experiments conducted in online settings, researchers employed much larger batches, with numbers ranging from 50 to 200 per user at a time. Some studies employ extreme multi-class experiments at a large scale (Cheng et al., 2016; Covington, Adams, & Sargin, 2016), using industrial hardware and software. Three factors limited the size of the set of classes. First, we did not have access to industrial-grade systems to train, validate, test, and deploy models at such an extreme scale. Second, during the pre-processing stage, we ran into memory issues when performing feature-engineering procedures. Initially, we had close to 2000 features for each user, with nearly half a million users. Manipulating such a large matrix became impossible given hardware's computational constraints. In that context, we could have used a larger set of apps to test the models because we had initially selected the top 500 most popular apps. In the end, we were limited to 100 app acquisition columns per user. Third, we still ran into issues when trying to score more than the ten most popular apps due to the extreme sparsity of the dataset we were working with. Another limitation is the fact we conducted our model deployment test in Spark was done on a VM rather than an actual Spark cluster. Although the software environment closely matches the real-world settings, we only employed a single CPU on a cluster with a single node. Therefore, we had to simulate distributed training by running several trials sequentially within the Spark environment. In future studies, the model deployment experiment should be extended to a real-world industrial cluster or at least a multi-node cluster in a laboratory setting. This will give us greater confidence in the quality of the results we have obtained while also contributing to the nascent academic research in the field of distributed GP-GPU and the efficient deployment of deep-learning models at a big data scale.

In the second experiment, we were able to obtain revenue benchmarks for the different recommender system environments using the novel evaluation framework. These obtained estimates were only partially validated using public earnings report information in the SVoD market. In the particular case of Netflix, the obtained revenue estimate using our method was close (with a difference of 0.47$), but the results for Hulu and HBO Max have larger gaps. These gaps might be explained by at least four factors: significant differences between the number of subscribers and viewers, unknown take rate for recommender systems in Hulu and HBO Max, unknown public information quality, and widely differing revenue models which might make it difficult to make valid comparisons across platforms.

In addition to the limitations related with each individual experiment, we can also add that in this work, we only considered the effects of the model's accuracy and efficiency, but it is known that the user interface plays a dominant role in the recommendations' success (Zheng, Wang, Zhang, Li, & Yang, 2010). No study to date has addressed the mobile app store interface's impact on the perceived quality of recommendations and what its relationship is with the model's financial impact. Our proposed metric can be used to implement a balanced scorecard approach to recommender system evaluation, integrating the UX and estimated profitability perspectives.

Additionally, as we discussed in the end of Section 3.1, we have some concerns that the assumption of the individual actions following a Poisson distribution might not hold in the current application. While we are confident that it should not greatly impact the results, we believe that a more realistic model could be obtained by modifying the NBD-Dirichlet to include non-independent actions in adjacent periods. We believe this could be addressed in future work.

CRediT authorship contribution statement

Roberto Henriques: Conceptualization, Validation, Writing – review & editing, Supervision. **Luís Pinto:** Conceptualization, Writing – original draft, Methodology.

Declaration of competing interest

The authors declare they have no conflict of interest.

Data availability

The data that has been used is confidential

Acknowledgments

We gratefully acknowledge the support of Aptoide in providing access to the data which made this project possible. This work was supported by national funds through FCT (Fundação para a Ciência e a Tecnologia), Portugal, under the project—UIDB/04152/2020—Centro de Investigação em Gestão de Informação (MagIC)/NOVA IMS.

References

Amazon Web Services (2018). Amazon EMR Pricing.

Anderson, C. (2008). *The long tail, revised and updated edition: Why the future of business is selling less of more* (p. 267). Hyperion.

Ansari, A., Li, Y., & Zhang, J. Z. (2018). Probabilistic topic model for hybrid recommender systems: A stochastic variational Bayesian approach. *Marketing Science*, *37*(6), 987–1008. http://dx.doi.org/10.1287/mksc.2018.1113.

Apache Foundation (2018). Apache Spark FAQ.

App Annie (2017). *App Annie app monetization report 2016: Technical report.*

App Annie (2018). *App Annie 2017 retrospective: Technical report*, (p. 162).

Batselier, J., & Vanhoucke, M. (2017). Improving project forecast accuracy by integrating earned value management with exponential smoothing and reference class forecasting. *International Journal of Project Management*, *35*(1), 28–43. http://dx.doi.org/10.1016/j.ijproman.2016.10.003.

Beel, J., & Langer, S. (2015). A comparison of offline evaluations, online evaluations, and user studies in the context of research-paper recommender systems. In S. Kapidakis, C. Mazurek, & M. Werla (Eds.), *Research and advanced technology for digital libraries* (pp. 153–168). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-24592-8_12.

Bilgihan, A., Kandampully, J., & Zhang, T. C. (2016). Towards a unified customer experience in online shopping environments: Antecedents and outcomes. *International Journal of Quality and Service Sciences*, *8*(1), 102–119. http://dx.doi.org/10.1108/IJQSS-07-2015-0054.

Blattberg, R. C., Kim, B.-D., & Neslin, S. A. (2008). *Database marketing - analyzing and managing customers*. Springer Berlin Heidelberg.

Bound, J. A. (2009). The S parameter in the Dirichlet-NBD model : A simple interpretation. *Journal of Empirical Generalisations in Marketing Science*, *12*(3), 1–7.

Brynjolfsson, E., Hu, Y. J., & Simester, D. (2011). Goodbye Pareto principle, hello long tail: The effect of search costs on the concentration of product sales. *Management Science*, *57*(8), 1373–1386. http://dx.doi.org/10.1287/mnsc.1110.1371.

Brynjolfsson, E., Hu, Y. J., & Smith, M. D. (2006). From Niches to Riches: Anatomy of the long tail. *Sloan Management Review*, *47*(4), 67. http://dx.doi.org/10.2139/ssrn.918142.

Bytheway, A. (2014). *Investing in information - The information management body of knowledge*. Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-11909-0.

Carson, J., & Brady, B. (2021). *The year in streaming - 2021: Technical report*, Antenna, URL https://puck.news/wp-content/uploads/2022/03/2021-Antenna-Year-in-Streaming.pdf.

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., et al. (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems* (pp. 7–10). http://dx.doi.org/10.1145/2988450.2988454.

Choi, H., & Mela, C. F. (2019). Monetizing online marketplaces. *Marketing Science*, (November), http://dx.doi.org/10.1287/mksc.2019.1197, p. mksc.2019.1197.

Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM conference on recommender systems - RecSys '16* (pp. 191–198). http://dx.doi.org/10.1145/2959100.2959190.

Curry, D. (2023a). HBO max revenue and usage statistics (2023). URL https://www.businessofapps.com/data/hbo-max-statistics/.

Curry, D. (2023b). Hulu revenue and usage statistics (2023). URL https://www.businessofapps.com/data/hulu-statistics/.

Dawes, J., Meyer-Waarden, L., & Driesener, C. (2015). Has brand loyalty declined? A longitudinal analysis of repeat purchase behavior in the UK and the USA. *Journal of Business Research*, *68*(2), 425–432. http://dx.doi.org/10.1016/j.jbusres.2014.06.006.

Ehrenberg, A., & Goodhardt, G. (2002). Double Jeopardy revisited, again. *Marketing Research*, *14*(1), 40–42. http://dx.doi.org/10.2307/1251818.

Ehrenberg, A. S. C., Goodhardt, G. J., & Barwise, T. P. (1990). Double Jeopardy revisited. *Journal of Marketing*, *54*(3), 82. http://dx.doi.org/10.2307/1251818.

Ehrenberg, A. S., Uncles, M. D., & Goodhardt, G. J. (2004). Understanding brand performance measures: Using Dirichlet benchmarks. *Journal of Business Research*, *57*(12 SPEC.ISS.), 1307–1325. http://dx.doi.org/10.1016/j.jbusres.2002.11.001.

Fader, P., Hardie, B., & Berger, P. D. (2004). Customer-base analysis with discrete-time transaction data. *SSRN Electronic Journal*, http://dx.doi.org/10.2139/ssrn.596801.

Fader, P. S., Hardie, B. G. S., & Lee, K. L. (2005). "Counting Your Customers" the easy way: An alternative to the Pareto/NBD Model. *Marketing Science*, *24*(2), 275–284. http://dx.doi.org/10.1287/mksc.1040.0098.

Farnum, N. R., & Stanton, L. W. (1987). Some results concerning the estimation of beta distribution parameters in PERT. *Journal of the Operational Research Society*, *38*(3), 287. http://dx.doi.org/10.2307/2581949, URL https://www.jstor.org/stable/2581949?origin=crossref.

Fleder, D. M., & Hosanagar, K. (2007). Recommender systems and their impact on sales diversity. In *Proceedings of the 8th ACM conference on electronic commerce* (p. 192). http://dx.doi.org/10.1145/1250910.1250939.

Fleder, D., & Hosanagar, K. (2009). Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management Science*, *55*(5), 697–712. http://dx.doi.org/10.1287/mnsc.1080.0974, URL http://pubsonline.informs.org/doi/abs/10.1287/mnsc.1080.0974.

Frank Robert, H., & Cook Philip, J. (1995). *The winner-take-all-society*. Penguin.

Fuhrer, B. (2022). Streaming grabs 35% of TV time in August, but overall usage dips as summer winds down. *Technical report*, Nielsen.

Fujitsu (2017). A reference model for high performance data analytics(HPDA) using an HPC infrastructure. *Technical report*, (pp. 1–8).

Georgiev, D. (2023). 17 Hulu statistics. URL https://techjury.net/blog/hulu-statistics/.

Ghoshal, A., Kumar, S., & Mookerjee, V. (2015). Impact of recommender system on competition between personalizing and non-personalizing firms. *Journal of Management Information Systems*, *31*(4), 243–277. http://dx.doi.org/10.1080/07421222.2014.1001276.

Gilotte, A., Calauzènes, C., Nedelec, T., Abraham, A., & Dollé, S. (2018). Offline A/B testing for recommender systems. In *Proceedings of the eleventh ACM international conference on web search and data mining* (pp. 198–206). http://dx.doi.org/10.1145/3159652.3159687.

Goel, S., Broder, A., Gabrilovich, E., & Pang, B. (2010). Anatomy of the long tail : Ordinary people with extraordinary tastes. *Business*, *48*(5), 201–210. http://dx.doi.org/10.1145/1718487.1718513.

Gomez-Uribe, C. A., & Hunt, N. (2016a). The Netflix recommender system. *ACM Transactions on Management Information Systems*, *6*(4), 1–19. http://dx.doi.org/10.1145/2843948.

Gomez-Uribe, C. A., & Hunt, N. (2016b). The Netflix recommender system. *ACM Transactions on Management Information Systems*, *6*(4), 1–19. http://dx.doi.org/10.1145/2843948, URL https://dl.acm.org/doi/10.1145/2843948.

Goodhardt, G. J., Ehrenberg, A. S. C., & Chatfield, C. (1984). The Dirichlet : A comprehensive model of buying behaviour. *Journal of the Royal Statistical Society*, *147*(5), 621–655. http://dx.doi.org/10.2307/2981696.

Graham, C., Bennett, D., Franke, K., Henfrey, C. L., & Nagy-Hamada, M. (2017). Double Jeopardy – 50 years on. Reviving a forgotten tool that still predicts brand loyalty. *Australasian Marketing Journal*, *25*(4), 278–287. http://dx.doi.org/10.1016/j.ausmj.2017.10.009.

Gupta, S., Hanssens, D., Hardie, B., Kahn, W., Kumar, V., Lin, N., et al. (2006). Modeling customer lifetime value. *Journal of Service Research*, *9*(2), 139–155. http://dx.doi.org/10.1177/1094670506293810.

Han, S. C., Lim, T., Long, S., Burgstaller, B., & Poon, J. (2021). GLocal-K: Global and local kernels for recommender systems. In *International Conference on Information and Knowledge Management, Proceedings: vol. 1*, (no. 1), (pp. 3063–3067). Association for Computing Machinery, http://dx.doi.org/10.1145/3459637.3482112, arXiv:2108.12184.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, *22*(1), 5–53. http://dx.doi.org/10.1145/963770.963772.

Huang, C.-Y. (2011). Excess loyalty in online retailing. *International Journal of Electronic Commerce*, *16*(2), 115–134. http://dx.doi.org/10.2753/JEC1086-4415160206.

Hunter, T. (2016). *Deep learning with apache spark and TensorFlow: Technical report.*

Iwata, T., Saito, K., & Yamada, T. (2008). Recommendation method for improving customer lifetime value. *IEEE Transactions on Knowledge and Data Engineering*, *20*(9), 1254–1263. http://dx.doi.org/10.1109/TKDE.2008.55.

Jannach, D., & Jugovac, M. (2019). Measuring the business value of recommender systems. *ACM Transactions on Management Information Systems*, *10*(4), 1–23. http://dx.doi.org/10.1145/3370082, URL https://dl.acm.org/doi/10.1145/3370082, http://arxiv.org/abs/1908.08328, arXiv:1908.08328.

Jeuland, A. P., Bass, F. M., & Wright, G. P. (1980). A multibrand stochastic model compounding heterogeneous erlang timing and multinomial choice processes. *Operations Research*, *28*(2), 255–277. http://dx.doi.org/10.1287/opre.28.2.255.

Ju, J. Y., Choi, I. Y., Kim, J. K., & Moon, H. S. (2017). Reinforcement learning for profit maximization of recommender systems reinforcement learning for profit maximization of recommender systems completed research paper. In *Proceedings of the P Re-ICIS 2017 S IGDSA symposium*. Seoul.

Kang, W.-C., & McAuley, J. (2018). Self-Attentive Sequential Recommendation. In *IEEE international conference on data mining: vol. 2018-Novem*, (pp. 197–206). URL http://arxiv.org/abs/1808.09781, arXiv:1808.09781.

Karsmakers, P., Pelckmans, K., & Suykens, J. A. K. (2007). Multi-class kernel logistic regression : A fixed-size implementation. In *Advances in neural information processing systems: vol. 20*, (pp. 1177–1184).

Katsov, I. (2018). *Introduction to algorithmic marketing - Artificial intelligence for marketing operations*. Grid Dynamics.

Kim, D., & Suh, B. (2019). Enhancing VAEs for collaborative filtering. In *Proceedings of the 13th ACM conference on recommender systems* (pp. 403–407). New York, NY, USA: ACM, http://dx.doi.org/10.1145/3298689.3347015, URL https://dl.acm.org/doi/10.1145/3298689.3347015.

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. In *ICLR 2015* (pp. 1–15). http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503.

Kiran, R., Kumar, P., & Bhasker, B. (2020). DNNRec : A novel deep learning based hybrid recommender system. *Expert Systems with Applications*, *144*, Article 113054. http://dx.doi.org/10.1016/j.eswa.2019.113054.

Knijnenburg, B. P., Willemsen, M. C., Gantner, Z., Soncu, H., & Newell, C. (2012). Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction*, *22*(4–5), 441–504. http://dx.doi.org/10.1007/s11257-011-9118-4, arXiv:z0022.

Lees, G. (2009). Are radio markets Dirichlet ? A study into the Nbd / Dirichlet , its empirical generalisations and their extension to radio.

Lu, F., Dumitrache, A., & Graus, D. (2020). Beyond optimizing for clicks: Incorporating editorial values in news recommendation. In *Proceedings of the 28th ACM conference on user modeling, adaptation and personalization* (pp. 145–153). New York, NY, USA: ACM, http://dx.doi.org/10.1145/3340631.3394864, URL https://dl.acm.org/doi/10.1145/3340631.3394864, http://arxiv.org/abs/2004.09980, arXiv:2004.09980.

Lu, J., Karmarkar, U., & Venkatraman, V. (2019). Planning to binge: How consumers choose to allocate media viewing time. *SSRN Electronic Journal*, *4*(April), 1–55. http://dx.doi.org/10.2139/ssrn.3493755, URL https://www.ssrn.com/abstract=3493755.

Luaces, O., Díez, J., Barranquero, J., del Coz, J. J., & Bahamonde, A. (2012). Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence*, *1*(4), 303–313. http://dx.doi.org/10.1007/s13748-012-0030-x.

Moore, K. (2020). How long would it take to watch all of Netflix? URL https://www.whats-on-netflix.com/news/how-long-would-it-take-to-watch-all-of-netflix/.

Netflix (2022). *Netflix shareholder letter - 2021 Q4*: *Technical report*, URL http://q4live.s22.clientfiles.s3-website-us-east-1.amazonaws.com/959853165/files/doc_financials/2021/q4/FINAL-Q4-21-Shareholder-Letter.pdf.

Obedkov, E. (2022). Steam games revenue grew to $3.1 billion in first half of 2022, but analysts expect slight recession.

Park Associates and Symphony Media A. I. (2021). AI-enabled data : Key to video service optimization AI-enabled data : Key to. *Technical report*, Park Associates, URL https://www.parksassociates.com/whitepapers/ai-wp2021.

Pathak, B., Garfinkel, R., Gopal, R. D., Venkatesan, R., & Yin, F. (2010). Empirical analysis of the impact of recommender systems on sales. *Journal of Management Information Systems*, *27*(2), 159–188. http://dx.doi.org/10.2753/MIS0742-1222270205.

Peltier, S., & Moreau, F. (2012). Internet and the 'long tail versus superstar effect' debate: Evidence from the French book market. *Applied Economics Letters*, *19*(8), 711–715. http://dx.doi.org/10.1080/13504851.2011.597714.

Prechelt, L. (2012). Early stopping — But when? In *Neural Networks: Tricks of the Trade* (pp. 53–67). http://dx.doi.org/10.1007/978-3-642-35289-8{_}5.

Pressman, L. (2015). Netflix statistics: How many hours does the catalog hold? URL https://automatedinsights.com/blog/netflix-statistics-how-many-hours-does-catalog-hold/.

Project Management Institute (2017). *A guide to the project management body of knowledge (PMBOK Guide)* (p. 763). Project Management Institute.

Pu, P., & Chen, L. (2010). A user-centric evaluation framework of recommender systems. In *CEUR Workshop Proceedings*: *vol. 612*, (pp. 14–21). http://dx.doi.org/10.1145/2043932.2043962.

Quan, T. W., & Williams, K. R. (2017). Product variety, across-market demand heterogeneity, and the value of online retail. *SSRN Electronic Journal*, (2054), http://dx.doi.org/10.2139/ssrn.2993236.

Rahimi, A., & Recht, B. (2007). Random features for large-scale kernel machines. In *NIPS'07, NIPS'07: Proceedings of the 20th International Conference on Neural Information Processing Systems* (p. 1177–1184). Red Hook, NY, USA: Curran Associates Inc..

Rayburn, D. (2022). A list of the latest ARPU (Average Revenue Per User) numbers from streaming services. URL https://www.streamingmediablog.com/.

Rosen, S. (1981). The economics of superstars. *The American Economic Review*, *71*(5), 845–858.

Rungie, C. (2003). How to estimate the parameters of the Dirichlet model using likelihood theory in excel. *Marketing Bulletin*, *14*, *Techni*(January 2003), 1–9.

Schmittlein, D. C., Morrison, D. G., & Colombo, R. (1987). Counting your customers: Who-Are they and what will they do next? *Management Science*, *33*(1), 1–24. http://dx.doi.org/10.1287/mnsc.33.1.1.

Schnabel, T., Bennett, P. N., & Joachims, T. (2018). Improving recommender systems beyond the algorithm. arXiv preprint arXiv:1802.07578.

Statista (2018). *Distribution of free and paid apps in the Apple App Store and Google Play as of 1st quarter 2018*: *Technical report*.

Statista (2022a). *Average daily time spent watching TV in the United States from 2019 to 2023*: *Technical report*, URL https://www.statista.com/statistics/186833/average-television-use-per-person-in-the-us-since-2002/.

Statista (2022b). *Gross income generated by Epic Games worldwide from 2018 to 2025*: *Technical report*, URL https://www.statista.com/statistics/1234193/epic-games-annual-income/.

Statista (2023a). *Number of Hulu's paying subscribers in the United States from 1st quarter 2019 to 1st quarter 2023*: *Technical report*, URL https://www.statista.com/statistics/258014/number-of-hulus-paying-subscribers/.

Statista (2023b). *Number of Netflix paid subscribers worldwide from 1st quarter 2013 to 4th quarter 2022*: *Technical report*, URL https://www.statista.com/statistics/250937/quarterly-number-of-netflix-streaming-subscribers-in-the-us/{#}:{~}:text=Netflixreported74.3millionpaid,quarteroftheprevioussyear.

Stocchi, L., Guerini, C., & Michaelidou, N. (2017). When are apps worth paying for? *Journal of Advertising Research*, *57*(3), 260–271. http://dx.doi.org/10.2501/JAR-2017-035.

Susic, P. (2023). HBO max subscriber numbers. URL https://headphonesaddict.com/hbo-max-subscribers/.

Tsoumakas, G., & Vlahavas, I. (2007). Random k-labelsets: An ensemble method for multilabel classification. In *Machine learning: ECML 2007* (pp. 406–417). Berlin: Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-540-74958-5{_}38.

van Hasselt, H. (2013). Estimating the maximum expected value: An analysis of (Nested) cross validation and the maximum sample average. (pp. 1–17). arXiv preprint arXiv:1302.7175.

Vanderlooy, S., & Hüllermeier, E. (2008). A critical analysis of variants of the AUC. *Machine Learning*, *72*(3), 247–262. http://dx.doi.org/10.1007/s10994-008-5070-x.

Verbeke, W., Dejaeger, K., Martens, D., Hur, J., & Baesens, B. (2012). New insights into churn prediction in the telecommunication sector: A profit driven data mining approach. *European Journal of Operational Research*, *218*(1), 211–229. http://dx.doi.org/10.1016/j.ejor.2011.09.031.

Wang, K., & Khan, M. M. H. (2015). Performance prediction for apache spark platform. In *2015 IEEE 17th International conference on high performance computing and communications, 2015 IEEE 7th international symposium on cyberspace safety and security, and 2015 IEEE 12th international conference on embedded software and systems*. http://dx.doi.org/10.1109/HPCC-CSS-ICESS.2015.246.

Weinhardt, C., Anandasivam, A., Blau, B., Borissov, N., Meinl, T., Michalk, W., et al. (2009). Cloud computing – A classification, business models, and research directions. *Business & Information Systems Engineering*, *1*(5), 391–399. http://dx.doi.org/10.1007/s12599-009-0071-2.

Wilson, L. (2018). A Complete guide to app store optimization (ASO). *Search Engine Journal*.

Wise, J. (2022a). Epic games store statistics 2022: How many users does it have? URL https://earthweb.com/epic-games-store-statistics/.

Wise, J. (2022b). How many people use steam in 2022? URL https://earthweb.com/how-many-people-use-steam/.

Wright, M., Sharp, A., & Sharp, B. (2002). Market statistics for the Dirichlet model : Using the juster scale to replace panel data. *International Journal of Research in Marketing*, *19*(1), 81–90. http://dx.doi.org/10.1016/S0167-8116(02)00049-6.

Wrigley, N., & Dunn, R. (1985). Stochastic panel-data models of urban shopping behavior .4. incorporating independent variables into the Nbd and Dirichlet models. *Environment and Planning A*, *17*(3), 319–331.

Yin, H., Cui, B., Li, J., Yao, J., & Chen, C. (2012). Challenging the long tail recommendation. *Proceedings of the VLDB Endowment*, *5*(9), 896–907. http://dx.doi.org/10.14778/2311906.2311916.

Zhang, Y., Bradlow, E. T., & Small, D. S. (2015). Predicting customer value using clumpiness: From RFM to RFMC. *Marketing Science*, *34*(2), 195–208. http://dx.doi.org/10.1287/mksc.2014.0873, URL http://pubsonline.informs.org/doi/10.1287/mksc.2014.0873.

Zhang, F., Liu, Q., & Zeng, A. (2017). Timeliness in recommender systems. *Expert Systems with Applications*, *85*, 270–278. http://dx.doi.org/10.1016/j.eswa.2017.05.038.

Zheng, H., Wang, D., Zhang, Q., Li, H., & Yang, T. (2010). Do clicks measure recommendation relevancy? In *Proceedings of the fourth ACM conference on recommender systems - RecSys '10* (p. 249). http://dx.doi.org/10.1145/1864708.1864759.

Zhong, N., & Michahelles, F. (2012). Where should you focus: Long tail or superstar? In *SIGGRAPH Asia 2012 Symposium on Apps on - SA '12*: *vol. 3*, (p. 1). New York: ACM Press, http://dx.doi.org/10.1145/2407696.2407706.

Zhong, N., & Michahelles, F. (2013). Google play is not a long tail market: An empirical analysis of app adoption on the Google play app market. *Proceedings of the 28th Annual ACM*, 499–504. http://dx.doi.org/10.1145/2480362.2480460.

Zhu, J., & Hastie, T. (2005). Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, *14*(1), 185–205. http://dx.doi.org/10.1198/106186005X25619.