POLITECNICO MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

User-centric vs. System-centric Evaluation of Recommender Systems: A Case of Study

Tesi di Laurea Magistrale in Computer Science Engineering
Ingegneria Informatica

Author: **Costantino Lo Bello**

Student ID: 953517
Advisor: Prof. Paolo Cremonesi
Co-advisors: Davide Frey
Academic Year: 2021-2022

Abstract

Recommender Systems (RS) are software tools that aim to reduce the information overload on the web by proposing possibly interesting items to the user. RS are widely used in many application domains such as e-commerce, tourism and movie recommendation. There are two main approaches to evaluating the quality of RS. System-centric, also called offline, is based on datasets of preferences and opinions on items previously collected from users, thus inexpensive and easy to reproduce. The other approach, called user-centric, also called online, measures the quality of the RS when real users interact with the system; it is an expensive approach to execute but leads to significant results. However, many works concluded that the results of the two approaches are often not correlated. We worked with Blacknut, a videogame startup company, to verify whether, in this particular domain, the two previously mentioned approaches correlate or not. As a methodological approach, we performed two studies to accomplish our goal. The first was a system-centric study based on accuracy error metrics and classification metrics to select promising candidates for the online test. The second was a user-centric study; we opted to perform an A/B test, the computer engineering version of randomized controlled trials. Different sets of users test different recommendation algorithms, providing performance measures by analyzing system logs. We concluded that accuracy error metrics are misleading in predicting the online performances of the algorithms. Instead, the ranking predicted through accuracy classification was reflected in the results of the user-centric study. The findings of this work enlarge the datasets of studies that compare the system-centric and the user-centric approaches and can also be used to design an RS in this domain. Assessing an RS's quality is an important open question; measuring the system's quality before doing expensive online experiments would be an important resource for the companies.

Keywords: Recommender Systems, Evaluation, A/B Testing, User study

Abstract in lingua italiana

I Sistemi di Raccomandazione sono strumenti software in grado di ridurre il sovraccarico di informazioni nel web proponendo oggetti possibilmente interessanti all'utente. I sistemi di raccomandazione sono vastamente utilizzati in molti settori come l'e-commerce, il turismo e servizi di streaming musicali. Ci sono due approcci principali per valutare la qualità di un sistema di raccomandazione. System-centric, anche chiamato offline, basato su dataset di preferenze e opinioni su oggetti precedentemente collezionati sugli utenti, quindi poco costoso e facile da riprodurre. L'altro metodo chiamato user-centric, anche detto online, misura la qualità del sistema di raccomandazione quando veri utenti interagiscono con il sistema; è un metodo costoso da eseguire ma porta a risultati significativi. Tuttavia, molte ricerche hanno concluso che i due approcci precedentemente descritti non sono sempre correlati. Abbiamo lavorato con Blacknut, una startup di videogiochi, per verificare se in questo particolare dominio i due approcci di valutazioni sono correlati o meno. Come approccio metodologico, abbiamo eseguito due studi per raggiungere il nostro obiettivo. Il primo è stato uno studio system-centric basato su accuracy error metrics e classification error metrics per selezionare candidati promettenti da testare nel test online. Il secondo è stato uno studio user-centric; abbiamo optato per svolgere un A/B test, la versione informatica di studi randomizzati controllati. Differenti gruppi di utenti provano differenti algoritmi di raccomandazione, la performance finali sono valutate analizzando i log del sistema. Abbiamo concluso che le accuracy error metrics sono erronee nel predire le performance online degli algoritmi. Al contratrio, la classifica predetta dalle accuracy clasification metrics rispecchia le performance ottenute dagli algoritmi online. I risultati ottenuti in questa ricerca vanno ad arricchire la collezione di studi che comparano i metodi di valutazione system-centric con quelli user-centric e posso essere utilizzati per sviluppare un sistema di raccomandazione in questo settore, cioè del videogame. Predire le performance di un algoritmo di raccomandazione offline è un'importante domanda aperta, essere capaci di valutare la qualità di un algoritmo senza costosi esperimenti online sarebbe un'importante risorsa per le aziende.

Parole chiave: Sistemi di Raccomandazione, Valutazione, A/B Testing, User study

Contents

1 | Introduction

Recommender system have found application in a variety of online settings including e-commerce, social networks or news websites. As a result both academia and industry have proposed a variety of algorithms in an effort to provide better quality recommendations. Two families of techniques exist to evaluate the quality the of recommendation [8, 35]: user-centric (online) and system-centric (online).

Online/user-centric evaluation consists in directly measuring the effect of a deployed recommender system in terms of click rate, sales, or other application-specific metrics. This can be achieved through A/B testing[10, 11] the computer-engineering version of randomized controlled trials, different sets of users unknowingly test different recommendation algorithms, providing performance measures in terms of click/purchase rates. Another possible way to performer user-centric evaluation is trough user-studies, in short lab experiments on which users are gather and they are asked to interact with the running recommender, after a period of testing users are eventually asked to answer a questionnaire to measure their satisfaction. While effective, online evaluation can be very costly and above all requires the ability to tweak a running recommender system. As a result, it is often a prerogative of large companies that need to evaluate their own recommender systems.

On the other hand, offline/system-centric evaluation computes recommendations on an offline dataset in a cheap and fast way. Historically Recommender Systems are often evaluated using accuracy metrics. Early Recommender Systems were often evaluated using error metrics such as RMSE (root mean squared error) or MAE (mean average error), while more recent research favors accuracy metrics such as precision, recall. Not only do these families of metrics offer diverging perspectives, but even the results of a single metric differ depending on the evaluation protocol employed in the evaluation [12, 33].

Unfortunately, many works have shown the predicted ranking between algorithms using system-centric evaluation does not translate in the same way in real production settings. Finding the best way to evaluate a recommender system without access to a production environment therefore remains an important open research question.

1.1. Contribution and goal

This thesis aims to enlarge the collection of studies that compare system-centric and user-centric evaluations to see if it is possible to establish a benchmark between offline evaluation based on accuracy metrics and online performances in the real world. Mind that this is an evaluation project, we are not proposing any new algorithm. To accomplish so, we worked with Blacknut, a cloud gaming company. We first managed to provide them with a new recommendation service used for our work to explore whether existing offline metrics can predict the impact of a recommendation algorithm online.

So the main research question that we will address is: *In a video game platform, how is the online performance of the recommender systems related to the performance of such systems measured in terms of their accuracy?*

1.2. Outline

The Thesis is structured as follow:

Chapter 2: provides an overview of the recommendation technique used during the work to give some basics to the reader in case of necessity.

Chapter 3: gives a description of the state of the art of the evaluation of Recommender Systems. Followed by a review of related works to our research.

Chapter 4: contains information about the context we are working in, and the instruments and functionalities used in our study.

Chapter 5: contains the proposed methodology and explanation of the design choices of our study. Followed by the actual execution of the experiment and their results. Finally a section containing the limitations of our experiment.

Chapter 6: conclusions and possible future developments.

2 | Algorithms

This chapter is a basic introduction to the most famous families of recommender algorithms. It aims to give a basic knowledge to understand the work that will follow and whose aim is to evaluate recommender systems.

2.1. Recommender Algorithms

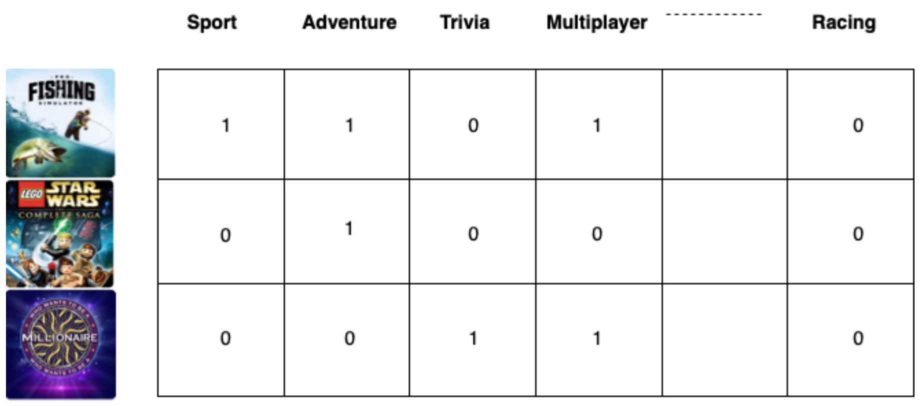
2.1.1. Input of Recommender

From a high-level point of view, a Recommender System has multiple sources of getting the information needed to generate recommendations; as the number of data increases, the complexity of the algorithms increases. First, the system needs to have information about the items in the catalogue and their attributes. Other valuable information is the data about the users, for example, their age or nationality. But the most crucial aspect is the interaction between the users and the items in the catalogue. Receiving feedback on what the user likes it's the most powerful resource for a recommender system.
More formally, we can define the actual inputs of recommender systems. We start with ICM[1] (Figure 2.1), in which the rows are the items in the catalogue, for instance in a video game recommender the items in the catalogue are the games that will be recommended, and columns are the attributes that can be the type of game such as adventure or sport. The other main input of a recommender system is the URM[2] (Figure 2.2), in which the rows are the users and columns are the ratings. Ratings can be explicit, for instance when a users directly rates a movie that he watched, or implicit, if a user watched a Youtube video until the end we can assume that as a positive rating.

2.1.2. Taxonomy

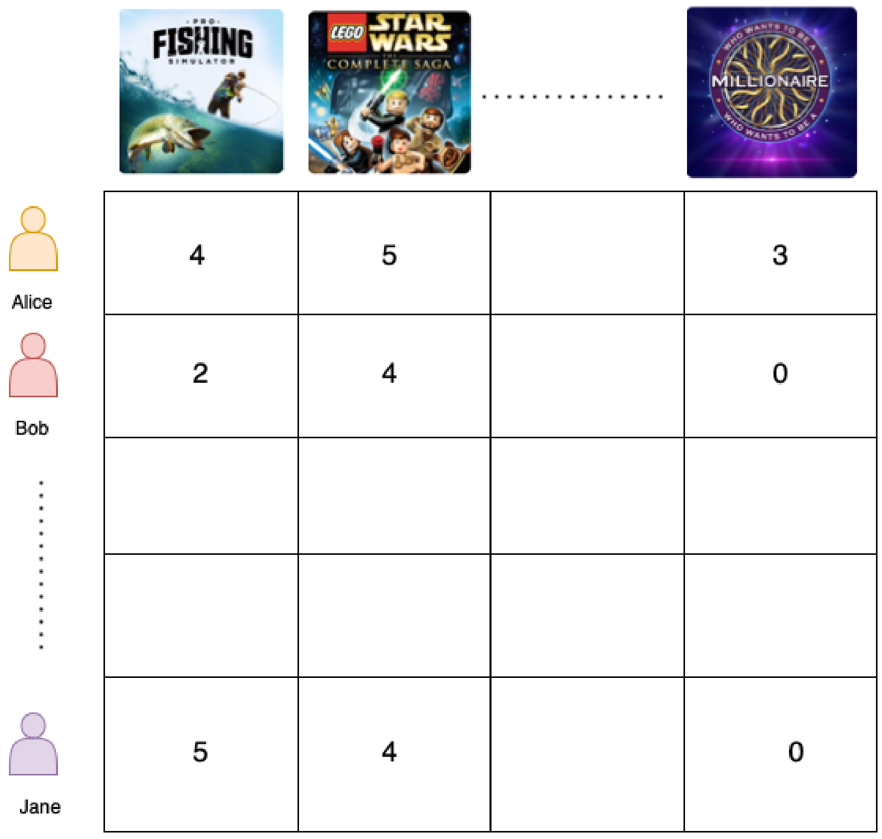
The first distinction between recommender techniques is between non-personalized and personalized algorithms. Non-personalized techniques are straightforward and consist

[1]Item-Content Matrix
[2]User-Rating Matrix



	Sport	Adventure	Trivia	Multiplayer	-----------	Racing
FISHING	1	1	0	1		0
STAR WARS	0	1	0	0		0
MILLIONAIRE	0	0	1	1		0

Figure 2.1: ICM



	FISHING	STAR WARS	....	MILLIONAIRE
Alice	4	5		3
Bob	2	4		0
⋮				
Jane	5	4		0

Figure 2.2: URM

in not tailoring the recommendations for a specific user, but all the users receive the same recommendations based on popularity, for instance. However, even if they are simple, they shouldn't be underestimated because they actually work well in practice; in Netflix homepage, the first kind of suggestion are the most popular movies of the week; in research, these techniques are an important baseline to compare against newly developed algorithms.

A more interesting family are the Personalized algorithms in which each user receives a tailored recommendation based on their interactions with the system. The main families of this category will be explained in the next sections.

2.1.3. Content Based Filtering

CBF[3] algorithms are based on the assumption that users who expressed a preference for an item will probably like similar items. For example, if a user likes a movie, he is likely to like another movie with the same attributes (e.g. leading actor). For doing so, the system needs to understand how similar the items are, based on their attributes. There are several formulas for measuring the similarities between two items [17]. This family of algorithms makes the recommendations with simple formula on which many other algorithms are based, using the weighted average of the previous rating given by the user:

$$p_{u,i} = \frac{\sum_{j \in I_u} r_{u,j} \cdot s_{i,j}}{\sum_{j \in I_u} s_{i,j}} \tag{2.1}$$

where $p_{u,i}$ is the predicated rating of the item, $s_{i,j}$ is the similarity between the item i and previously rated items j by the user u, and finally, $r_{u,j}$ is rating given by the user u to the item j.

2.1.4. Collaborative filtering

Collaborative filtering (CF) [27, 36] is a widely used approach to design recommender systems. It uses the past user-items interaction stored in the URM to generate recommendations. It is based on the assumption that users who give similar ratings to different items are likely to like similar items, so the algorithms can exploit the correlation between similar groups of users to estimate missing ratings. There are two leading families of CF algorithms, Memory-based and Model-based.

Memory based Collaborative Filtering

Memory based collaborative filtering algorithms, also referred to as neighborhood based, use users' ratings to compute similarity between users or items to provide recommendations. The ratings of users, or items, are predicted based on their neighborhood. Memory based algorithms can solve the recommendation problem with two approaches: user based collaborative filtering and item based collaborative filtering.

[3]Content Based Filtering

User-Based

In user based CF, the basic idea is that similar users rate the same item in similar ways. To predict the rating by user u of item i, a weighted average is performed over the ratings by users most similar to u on the same item. The similarities between u and the other users are used as weights.

$$r'_{u,i} = \frac{\sum_{v \in U} r_{v,i} \operatorname{sim}(u,v)}{\sum_{v \in U} \operatorname{sim}(u,v)} \tag{2.2}$$

where $\operatorname{sim}(u,v)$is the similarity between user u and user v.

Item-Based

In item based CF, the basic idea is that a user give similar ratings to similar items. While with user based the prediction of a rating is computed from the ratings given by other users on the same item, with item based, instead, the ratings of the target user on similar items are used, neighbours are defined by the similarity between items.

$$r'_{u,i} = \frac{\sum_{jinI_u} r_{u,i} \operatorname{sim}(i,j)}{\sum_{jinI_u} \operatorname{sim}(i,j)} \tag{2.3}$$

where $\operatorname{sim}(i,j)$ is the similarity between item $i$and $j$and I_u is the set of items rated by the user u.

Model-based

Model-based recommendation systems involve building a model based on the dataset of ratings. In other words, we extract some information from the dataset, and use that as a "model" to make recommendations without having to use the complete dataset every time.

2.1.5. Matrix factorization

This family of algorithm is based on latent factor that are attributes that tries to uncover features of observed ratings [24]. Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space. The latent space tries to explain ratings by characterizing both products and users on factors automatically inferred from user feedback. Accordingly, each item i is associated with a vector $\vec{y_i}$, and each user u is associated with a vector $\vec{x_i}$. For a given item-i, the elements of $\vec{y_i}$ measure the extent to which the item possesses those factors,positive or negative. For a given user u,the elements of $\vec{x_i}$ measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product, $\vec{x_i} \cdot \vec{y_i}$, captures the interaction between user u and item i,the user's overall interest in the item's characteristics. This approximates user u's rating of item i, which is denoted by r' , leading to the estimate:

$$r' = \vec{x_i} \cdot \vec{y_i} \tag{2.4}$$

To find the missing parameters the goal is minimizing the following error function with the addition of regularization term:

$$X^*, Y^* = \min \|R - R'\| + \lambda_a \|X\| + \lambda_b \|Y\| \tag{2.5}$$

in which the last two term are regularization factors to avoid overfitting and λ_a, λ_b are hyperparameters that need to be properly tuned.

Alternating Least Square

Alternating Least Square (ALS) is one of the most common techniques to solve the Matrix factorization problem, and it can be summarized as follow: after initializing the matrices X and Y with random values the steps are repeated until convergence:

1. Fix X and solve each row of V by treating the problem as least-square regression problem using only observed ratings

$$\sum_{u \in R_i} \left(R_{u,i} - \sum_{l=1}^{k} X_{u,l} Y_{i,l} \right)^2 \tag{2.6}$$

2. Fix Y and solve each row of by treating the problem as least-square regression problem using only observed ratings

$$\sum_{u \in R_i} \left(R_{u,i} - \sum_{l=1}^{k} X_{u,l} Y_{i,l} \right)^2 \tag{2.7}$$

2.1.6. SLIM

A way of design a model-based item-based collaborative filtering to perform a top-n recommendation task using machine learning techniques[29]. To estimate the ratings on an unseen item we recall the formula previously explained:

$$p_{u,i} = \sum_{j \in I_u} r_{u,j} \cdot s_{i,j} \tag{2.8}$$

In matrix form is:

$$\mathbf{R}' = \mathbf{R} \cdot \mathbf{S} \tag{2.9}$$

Our goal is to estimate the matrix S that contains the coefficient to learn thus the model, to do so we need to define a loss function over the observed rating and minimize it:

$$E(S) = (\|R - R \cdot S\|)_2 + \lambda_a \|S\| + \lambda_b \|S\| \tag{2.10}$$

subject to the constraints $S \geq 0$,and to regularization terms to avoid producing negative ratings, and $\text{diag}(S) = 0$, to avoid a solution in which S is the identity matrix.

3 | Evaluation

Evaluating the quality of an algorithm is a crucial step in any development project, to understand how well it performs and what can be improved. In particular this process is not trivial in the field of Recommender systems, this is because as many works have shown is very domain dependent [2, 6]. There are two main approaches to evaluate a Recommender System [6, 8, 18]:

- *System-centric* also called Offline evaluation: quality measures are evaluated using datasets of preferences and opinions on items previously collected from users that are not interacting with the RS under study.
- *User-centric* also called Online evaluation: users interact with a running recommender system and receive recommendations. Measures are collected by asking the user (e.g., through interviews or surveys), observing her behavior during use, or automatically recording interactions and then subjecting system logs to various analyses.

In the next sections of the chapter it will be presented the current state of the art of the various evaluation techniques and finally a comparison between them will be presented.

3.1. System-centric

Offline evaluation is broadly studied in other fields such as machine learning and Information Retrieval; what will follow is focused on recommender systems, but many concepts apply to other fields. Evaluating a recommender system's quality before its deployment is a valuable resource. Offline metrics are attractive because they require no interactions with real users and thus allow us to compare a wide range of algorithms at a low cost. They are a powerful tool to filter a set of candidate algorithms or for parameter tuning for these reasons. In order to define offline evaluation we need to take into account four aspects:task, dataset, partitioning and the metrics [7, 18].
First, the task we are dealing with, it's important to distinguish between possible tasks because, in the offline evaluation, one size does not fit all; according to the task we are

dealing with, different approaches are more valuable than others. The two main tasks of recommendation algorithms are **rating prediction** in which the recommender system predicts the rating of items for a given user, and the second possible task is **top-n recommendation task** in which the goal is not to predict an accurate rating but to propose a list of n items that the user might like.
The second aspect we have to deal with is the dataset. When analyzing a recommendation algorithm, we are interested in its future performance on new data rather than its performance on past data. We must properly partition the original dataset into training and test subsets to test future performance and estimate the prediction error. The training data are used by one or more learning methods to train the model (i.e., an entity that synthesizes the behaviour of the data), and the test data are used to evaluate the quality of the model. The dataset is usually split according to one of the following methods [7, 34]:

- **Holdout** is a method that splits a dataset into two parts: a training set and a test set. These sets could have different proportions. In the setting of recommender systems the partitioning is performed by randomly selecting some ratings from all (or some of) the users. The selected ratings constitute the test set, while the remaining ones are the training set. This method is also called leave-k-out.

- **Leave-one-out** is a method obtained by setting $k = 1$ in the leave-k-out method. Given an active user, we withhold in turn one rated item. The learning algorithm is trained on the remaining data. The withheld element is used to evaluate the correctness of the prediction and the results of all evaluations are averaged in order to compute the final quality estimate. This method has some disadvantages, such as the overfitting and the high computational complexity.

- **m-fold cross-validation** is a simple variant of the holdout method. It consists in partitioning the dataset into m independent folds (so that folds do not overlap). In turn, each fold is used exactly once as test set and the remaining folds are used for training the model. This technique is suitable to evaluate the recommending capability of the model when new users (i.e., users do not already belonging to the model) join the system. By choosing a reasonable number of folds we can compute mean, variance and confidence interval.

Finally according to the task we are dealing with some metrics might be more suitable than others. There are tree main families of metrics error metrics classification metrics and ranking metrics. Evaluation metrics are widely studied, and they are the core part of an evaluation process and they deserve separated sections due to also their variety.

3.1.1. Error metrics

Error metrics are particularly important in a rating prediction task, where the actual predicted rating is used .Error metrics were very interesting to researchers during the 2000s mostly due to the Netflix competition, in which a prize money of one million dollar was awarded to algorithm that performed better while being evaluated with RMSE [3], that we are going to define. This family of metrics measures how close the recommender system's predicted ratings are close to the true user ratings. The easiest way to compare the two ratings is to compute the error[18, 35]:

$$e_{u,i} = |r_{u,i} - p_{u,i}| \tag{3.1}$$

This is the error for one user and one item,where $r_{u,i}$ is the true rating of the user u to the item i and $p_{u,i}$ is the predicted rating, T is the test set and N is the number of non-zero rating , we want it for all the items in the test set . For this reason MAE is the first simple error metric introduced:

$$\text{MAE} = \frac{\sum_{u,i \in T} |r_{u,i} - p_{u,i}|}{N} \tag{3.2}$$

A variant of MAE is MSE:

$$\text{MSE} = \frac{\sum_{u,i \in T} (r_{u,i} - p_{u,i})^2}{N} \tag{3.3}$$

almost identical in the quality of what we are measuring, but it is used more because it is easier to minimize. Minimizing the error is the objective of some algorithms. The most popular error metric is RMSE due to the previously mentioned Netlfix competition, defined as follow:

$$\text{RMSE} = \sqrt{\frac{\sum_{u,i \in T} (r_{u,i} - p_{u,i})^2}{N}} \tag{3.4}$$

When using error metrics the error is computed only on a portion of data on which the ratings are known the eventual goal of the recommender is to predict the rating for items not yet rated by the user. This implies the usage of strong assumption called MAR missing as random[27], that consists on the fact that we are assuming that unknown rating have the same probability distribution of known ratings. This is considered unrealistic because for example if we rate an object from an e-commerce web site it is likely that we are going to rate it in positive way since we bought it because we liked it. The same example can be done with a movie catalog, there are many movies that we don't like before watching them maybe because we don't like the type, the MAR assumption in this situation fails because the missing rating is not because we haven't seen the movie yet, but it's because we don't like it.

3.1.2. Classification metrics

Error metrics are not suited for all the contexts, take for instance an e-commerce web site where a list of items is proposed to the user, knowing the predicted rate is not important in this situation, finding good items is what the Recommender system should do. Campochiaro et al. in [4] argued that having algorithms with very low RMSE do not necessarily perform well in top-n recommendation task, that we need to refer to information retrieval metrics. Classification metrics measure the frequency of which a recommender system makes correct or incorrect decisions about whether an item is good. In order to evaluate a model we usually take into account the following measures[18, 35]:

- True Positives (TP): number of instances classified as belonging to class A that truly belong to class A;
- True Negatives (TN): number of instances classified as not belonging to class A and that in fact do not belong to class A;
- False Positives (FP): number of instances classified as class A but that do not belong to class A;
- False Negatives (FN): instances not classified as belonging to class A but that in fact do belong to class A.

Now we can define the following metrics:

$$\text{precision@k} = \frac{\text{TP}}{\text{FP} + \text{TP}} \tag{3.5}$$

$$\text{recall@k} = \frac{\text{TP}}{\text{TN} + \text{FN}} \tag{3.6}$$

Precision has an intuitive meaning: it measures the percentage of interesting items suggested to the users, with respect to the total number of suggested items k. While Recall measures the percentage of interesting items k suggested to the users, with respect to the total number of interesting items in the catalog. Precision and Recall can be combined in a single quantity, called F-measure:

$$\text{F-measure@k} = \frac{2 \cdot \text{precision@k} \cdot \text{recall@k}}{\text{precision@k} + \text{recall@k}} \tag{3.7}$$

Usually users express their preference by giving a rate from 1 to 5 (for instance), while using classification metrics we need to set a threshold to classify an element as relevant or not. This might cause problems because different users might have different rating scale, for a user a rate of 3 can be considered that they didn't like the item for others in a positive way instead.

3.1.3. Ranking metrics

Ranking metrics are an evolution of classification metrics where the task of the recommender is not only to propose a list of good items, but also to predict how much a user likes an item compared to the others, in other words proposing an ordered list of items to the user. Take for instance an accommodation booking service in which generally the best options are put at the top of the list.
Mean Average Precision (MAP) is one of the most used rank metrics [18]. It computes the overall precision of the system at different lengths of recommendation lists. MAP is computed as the arithmetic mean of the average precision over the entire set of users in the test set. Average precision for the top K recommendations, AP@k, is defined as follows:

$$\text{AP@k} = \frac{\sum_{k=1}^{N} \text{Precision@k} \cdot \text{Rel}(k)}{M} \tag{3.8}$$

where N is the number of relevant items and Rel(i) is 1 if the i-th recommended item is relevant, 0 otherwise. From this definition, we can compute MAP as:

$$\text{MAP@n} = \frac{\sum_{u \in U} \text{AP}_u\text{@n}}{U} \tag{3.9}$$

where U denotes the users in the test set.
Mean Reciprocal Rank (MRR) is a well known metric in information retrieval . This metric evaluates the results of a recommender based on the order of probability of correctness. It is defined as following:

$$\text{MRR} = \frac{\sum_{u \in U} \frac{1}{\text{rank}_u}}{U} \tag{3.10}$$

where rank_u is the position of the first relevant retrieved answer for user u.

3.1.4. Beyond accuracy metrics

The evaluation of recommender systems traditionally was focused on reaching the highest accuracy, but there has been some works towards evaluating other qualities such as Diversity, Serendipity, Coverage and Novelty. This is because the final goal of a recommender system is not just to be accurate, but to propose items more interesting and engaging as possible in other words **useful**, and depending on the system domain and the user's needs. For instance, when recommending music, it is not always desirable to recommend unknown or surprising artists, as it may be important to include artists the user is familiar with but has not listened to in a while. What will follow are qualitative definitions of the previous qualities without giving formulas on how to compute them [21, 37].

- **Diversity**: Ensuring that the list of retrieved documents covers a broad area of the information space increases the chance of satisfying the user's information need. In RS research, diversity is generally related to how different are items in a recommendation list for the user.
- **Coverage**: The coverage of a recommender system is a measure of the domain of items in the system over which the system can form predictions or make recommendations. It can be associated with two concepts. Prediction coverage,the percentage of the items for which the system is able to generate a recommendation. And catalogue coverage,the percentage of the available items which effectively are ever recommended to a user.
- **Serendipity**: Serendipity is a difficult to define because its core component is the element of surprise. Herlocker [18] informally defined a serendipitous recommendation as one that helps the user find a "surprisingly interesting item he might not have otherwise discovered". More formally Serendipity is composed by two main aspects, Unexpectedness the item is not yet discovered by the user, and Usefulness the item is interesting, relevant to the user.
- **Novelty**: novelty can be seen as the ability of the system to propose to the user unknown items. This definitions of novelty often overlaps with serendipity, we can say that the first includes the latter because it doesn't require the element of surprise, for instance if the recommender system suggests to the user a new music album of an artist the the user previously liked, this is a novel recommendation but not serendipitous because the artist was already discovered by the user.

3.2. User-centric

System-centric evaluation provides a grate framework for systematically evaluate the quality of a Recommender System, but may works have argued that performance of algorithms that did great in an offline evaluation scenario, did not translate when the system was deployed in a real production environment [9, 12, 14, 19, 33]. To truly evaluate the quality of RS online evaluation is needed, of course it has some drawbacks, due to its cost and effort to set it up. There are several techniques to perform user-centric evaluation, that is less standardized approach respect to the system-centric evaluation[35].

3.2.1. User study

User studies typically measure user satisfaction by monitoring how the user interact with the systems, asking direct feedback and trough questionnaires(section later). Users gathered for the experiment receive recommendations from different recommendation algorithm and they are asked to interact with the systems performing several tasks, after a period of testing results are evaluated.

3.2.2. Participants

Recruiting participants is a crucial task for user-centric studies. It's important to gather a reasonable large sample size to reach statistical significant results, but at the same time the quality of the sample shouldn't be neglected. The sample needs to be similar to what it,s expected to see in the real word, the participants should be unbiased on the goal of the experiment. It's a common practice to use crowd-sourcing platform like Amazon Mechanical Truck[1] to test the mock-up of the application and answer the questionnaire.

3.2.3. Questionnaires

Questionnaires are powerful tools in a user study.The questions asked can provide information about properties that are difficult to measure, such as user satisfaction or sense of risk. For example if we want to measure metrics such as serendipity and diversity, it's not really straight forward to compute them analytically, much simpler would by to ask directly the user if that item is considered serendipitous or diverse by him. There are several framework proposed on how to build a proper questionnaire [31].

3.2.4. A/B testing

Online controlled experiments also know as A/B testing are very effective way to compare the performance of recommender algorithms [15]. They are a standard evaluation procedure also in other fields especially in the web environment [23]. In the simplest controlled experiment, users are randomly exposed to one of two variants: Control (A), or Treatment (B), the control is the version of the system in use before the experiments,

[1]`https://www.mturk.com`

the treatment is the new version to evaluate. Based on observations collected, an overall evaluation criterion is derived for each variant. Now we define some key concept for online controlled experiment [22]:

- *Overall Evaluation Criterion*: a quantitative measure of the experiment's objective.
- *Factors*: a controllable experimental variable that is thought to influence the Overall Evaluation Criterion. For example an algorithm.
- *Variant*: a user experience being tested by assigning levels to the factors. So for example if a factor is an algorithm the variant are different variants of it, so different algorithms.
- *Experimental unit*: the entity over which metrics are calculated before averaging over the entire experiment for each variant. Sometimes called an item. The units are assumed to be independent.

On the web, the user is a common experimental unit, although some metrics may have user-day, user-session or page views as the experimental units. For any of these, randomization by user is preferred. It is important that the users receive a consistent experience throughout the experiment.

3.2.5. Reaching statistical significance

Before starting any kind of experiment it is necessary to perform an *Hypothesis* such as : "does algorithm A performs better than Algorithm B?". Once collected data, it is important to statistically test the formulated hypotheses and verify that results are statistically significant (they are not due to luck),generalization is a necessary step because the result must be significant also on unseen data not just the data used for the test [18, 32, 35]. Several procedures are used:

- A standard tool for significance testing is a significance level or p-value the probability that the obtained results were due to luck. Generally, we will reject the null hypothesis that algorithm A is no better than algorithm B if the p-value is above some threshold. That is, if the probability that the observed ranking is achieved by chance exceeds the threshold, then the results of the experiment are not deemed significant.
- The difference between more than two conditions can be tested with an ANOVA (Analysis of Variance). The ANOVA test produces an F- statistic; its p-value signifies evidence against the null hypothesis that the dependent variable has the same

value in all conditions[1].

3.3. Comparison system-centric vs user-centric

Offline evaluation is based on past data, and it doesn't need real-time user interaction, so it can be implemented at a low cost. It can quickly test and evaluate the performance of different kinds of recommendation algorithms and is also highly reproducible when the steps are correctly defined. But the disadvantages are that such experiments can usually be used in evaluating the prediction accuracy of the algorithms or Top- N precision of recommendation, and can do little in other evaluation criteria, it has been shown that offline performance does not always correlate with online performance (see section related works). This procedure is used more to filter out inappropriate algorithms leaving a more valuable set of algorithms to test for the next phases.
Online evaluation such as controlled experiments and user studies need live user interaction with the system, and they can are used to understand the actual impact of the recommendation on the overall systems [19]. But it has some disadvantages, mainly because of its high cost in fact, it requires a long period of experimentation or/and large sample size to reach statistical evidence. This might not be appreciated by all the stakeholders. Another disadvantage is the risk of miss interpretation of the result due to a mistake made during the set-up phase, such as a wrong sampling of the users [20]. The table 3.1 summarizes the the main advantages and disadvantages of the two types of evaluation.

	ADVANTAGES	DISADVANTAGES
ONLINE	Powerful because the feedback are from real users.	Expensive and not easy to interpret.
OFFLINE	Not expensive and easy to reproduce	Do not always correlate with the actual performance.

Table 3.1: Pros and Cons Offline vs Online

3.4. Related works

This section will provide an overview of some related works that try to compare *system-centric* and *user-centric* evaluation. Different studies led to different results; some studies concluded that there is a correlation between the two approaches others believe that there isn't. Many works in this field are carried out by using user studies since they are easier to set up in academic fields, A/B testing instead permits reaching more significant results but researches with this method are less likely to be published because companies are less keen to share those information since they very business related.

3.4.1. Studies

Jannach and Jugovac in [19] conducted a user study to assess the quality perception of different session-based recommendation algorithms in a realistic scenario, and to compare it with offline result. To do so they designed a between-subjects user study (N=250), where participants interacted with an online radio application that was developed for the experiment, they split the sample in five groups and each group was assigned to a different algorithm, at the end of the experiment participants (gathered on Amazon mechanical) were asked to answer a questionnaire tailored by them to measure the perceived quality they not only asked how relevant was the recommendations but also if it was serendipitous and diverse. The result showed the algorithm such as the one used by Spotify performed badly while evaluated offline (Precision and Recall) but was positively considered by the participants compensating the lack of accuracy by helping the user to discover new tracks.
Rossetti et al. in [33] proposed a novel comparison approach between an offline evaluation protocol and an online user study, their goal was to see if it is possible to use offline accuracy metrics to reproduce the ranking of the algorithms that comes out from the user study.They designed an a with-in user study in which they test the same algorithms on the same users both offline and online. In the first rounds they asked users(gathered with mail listing) to browse a web interface and give rating on items they liked, in the second round they asked the same users to evaluate the same algorithms by rating five recommendation each, by creating a list of 20 items, on average, and they mixed the list. Then with the data gathered from both round they computed precision on all items and on long tail item, and only for the online data they computed how useful a recommendation was by combining relevance and novelty. One of their main result is that offline precision measurement does not provide the same ranking of algorithms as online precision does.
Cremonesi et al. in [8] argued if the algorithms that perform best in terms of system-centric quality generate recommendations that provide the best effects on decision making, in an e-tourism environment. The system-centric evaluation was performed on a dataset of 210000 simulated users, recall and fallout were measured: The User-centric evaluation aimed to measures several qualities of the recommendations such as choice satisfaction, choice risk, elapsed time, perceived time. Users were spilt into groups and to each group an algorithm was assigned, afterwards participants played with the mock-application developed for the test with the task of assembling a trip. Finally the performance were analyzed trough system logs and a questionnaire was given to the 240 participants gath-

ered trough the university mailing list. Their results different from other domains showed that touristic products may exploit different aspects of recommender systems, and suggest the possibility to adopt system-centric evaluation techniques as a good approximation of the user experience.

Garcin et al. in [13, 14, 26] tested their new algorithm developed for news recommendations and performed a live analysis on it, to evaluate the performance of their algorithm and analyze the difference between online and offline evaluation. The online evaluation is split in two phases where we test different strategies against each other. The first phase consists of comparing a standard context-tree system against the baselines in terms of click-through rate and page views. The second phase investigates different versions of CT recommender systems [13]. The recommender algorithms together received 172'013 clicks on 10'653 stories in the first phase and 285'572 clicks on 10'830 stories in the second phase. The result shows that in the offline case the algorithm couldn't outperform a baseline most popular strategy because the recommendations do not directly influence the user. On the contrary the most popular strategy didn't perform well online because the most popular articles weren't considered interesting by the user since they were already in the front page however the CT algorithm had better results. Further investigations on this topic have been carried out by the same authors by taking different metrics such as Serendipity and Novelty into account.

Mogenet et al. in [28] reviewed a comparative offline and online performances of three recommendations models,they analyzed how the offline performance metrics correlate with online metrics to understand how an offline evaluation process can be leveraged to inform the decisions. Different offline metrics were computed in the offline evaluation phase, especially ranking metrics, precision and recall. The online evaluation was performed using A/B testing assigning a bucket of user to each model, as online metric they use *apply-rate*, the percentage of times a user applied for a job. Afterwards the computed Pearson and Spearman correlation coefficient [17], they concluded that those offline evaluation metrics are reliable enough to decide to not deploy the new models when the offline performances are significantly negative; and to deploy the new models when there is a positive impact on the offline metrics.

Peska and Vojtas in [30] worked on bridging the gap between offline and online evaluation, the domain of their work was a medium size travelled agency, they tested 800 variants of algorithms offline and filtered to the online phase only 12. The study design was as usual divided in two phases, the dataset of the offline phase was composed of user's visit of a period of 2 and a half years, after cleaning it contained 270k interactions made by 72k users, a huge variety of metrics were evaluated such error metrics, classification metrics, ranking metrics and also metrics different from relevance metrics such as novelty

and diversity. When selecting the algorithms for the next phase their main goal was to determine predictability of online result from online metrics, the strategy chosen to select the algorithms was to pick ,from each offline metric the tested, the worst and the best algorithm leaving only 12 for the online phase. The online phase was conducted using A/B testing for a period of one month on the production server of travel agency, each user was assigned to an algorithm based on his ID, during the on-line evaluation, they monitored which objects were recommended to the user, whether (s)he clicked on some of them and which objects (s)he visited,based on the collected data, they evaluated two metrics: click through rate (CTR) and visit after recommend rate (VRR). Results have indicated a positive correlation between offline ranking metrics and CTR and VRR and a negative correlation with novelty for less senior users, results are reversed for senior users, this might indicate that senior users prefer simpler suggestions.
Fazeli et al. in [12]claimed that in the domain of a social learning platform, accuracy metrics to predict user satisfaction is just the tip of the Iceberg. The study was designed in two parts, first a data-centric evaluation aimed to asses the performance of the candidate recommender algorithms in terms of accuracy information retrieval metrics such as precision and recall. The dataset used contains interaction coming from the learning platform and consist on 9117 events of 2567 users with 3392 objects. Many algorithms and configuration were tried in this phase mainly KNN algorithms and matrix factorization. The second part of the study is a user-centric study, designed as followed. First they let the participants (gathered trough crowd sourcing) interact with the system with a message "there is no recommendations for you today" in order to build the user profile. The in the second phase users were randomly split and assigned to a recommendation algorithm and they were exposed to recommendations. Finally the were asked to answer a questionnaire to evaluate the perceived quality of the recommendations by the users. The questionnaire was designed taking as quality indicators : Accuracy,Novelty,Diversity,Usefulness and Serendipity. After analysing the results they concluded that the user-centric evaluation does not confirm the result obtained by the traditional system-centric evaluation based on accuracy.

Authors	domain	metrics	type	results
Cremonesi et al. in [8]	e-tourism	recall and fallout	User-study	correlation between the 2 phases
Garcin et al. in [14]	news recommendation	precision	A/B testing	not correlation between the 2 phases
Mogenet et al. in [28]	Jobs (Indeed)	ranking metrics, precision, recall	A/B testing	offline evaluation is a good tool to filter out bad candidate
Peska and Vojtas in [30]	medium travel agency	ranking, accuracy, novelty, diversity	A/B testing	the correlation depends on the seniority of the users
Rossetti et al. in [33]	movies	precision	User study	offline and online precision are not correlated.
Ludewig and Jannach in [25]	music	Precision and Recall	User-study	not correlation between perceived quality and offline metrics
Fazeli et al. in [12]	learning platform	Precision and Recall	User-study	The result obtained with a system-centric evaluation do not reflect the user-centric one.

Table 3.2: Comparison of different studies.

4 | Instruments

This chapter gives information about the context we are working in, video game recommendations, and a high-level description of the Blacknut platform. Followed by the main functionalities of the system we used in this project.

4.1. What is Blacknut?

Blacknut is a start-up company that provides a video-game streaming service[1]. Blacknut subscribers can enjoy a selection of more than 500 video games, Figure 4.1 shows the homepage, on their TV screen, mobile device, or laptop without requiring expensive dedicated hardware. The video game runs on the cloud on Blacknut's servers and streams content directly to user devices (Figure 4.3). Before the deployment of the recommender system, users can manually browse a catalogue of over 500 video games (Figure 4.2). However, during this project, we managed to successfully deliver a working RS that the company was able to run and display recommendations to real subscribers to allow them to experience new content at each visit. The data gathered from the interactions between the users and the system was crucial to our analysis.

4.2. Functionalities

This section provides a description of the main functionalities of Blacknut and its Recommender System involved in this project. Both from a researcher and a user perspective.

4.2.1. Making Recommendations

The core functionalities of the system is making recommendations, in other words producing a list of interesting items for the user, based on the user's previous interactions as we talked in previous chapters. The algorithms implemented available are:

- **Random**: given a recommendation list of length N, provide N movies randomly

[1]`https://www.blacknut.com/en`

selected from Blacknut catalogue.

- **itemavg**: a simple method to consider the popularity of the items, given a recommendation list of N elements, provide N items with the highest average rating.
- **itemuseravg**: it's a variant of the previous algorithm but it adds to the prediction a bias which measures how different the average rating of this user is compared to the average rating in the whole dataset.
- **ubknn**: in the user-based version (denoted UBKNN) , a kNN matrix is built to find the k nearest users of each user u. Candidate items for recommendation are those liked by the neighbors of u but which are still unknown to u. They are given a predicted score, and the most relevant ones get recommended with a top-N approach.
- **ibknn**: it follows the same approach of the previous algorithm but it computes the similarity between items instead of users.
- **Matrix factorization**: matrix factorization is a class of collaborative filtering algorithms used in recommender systems. Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. Several approaches are available for decomposing the matrix.

The administrator of the system can decide which algorithm to run and to display. The output of the recommender is eventually saved in an external table managed by Blacknut.

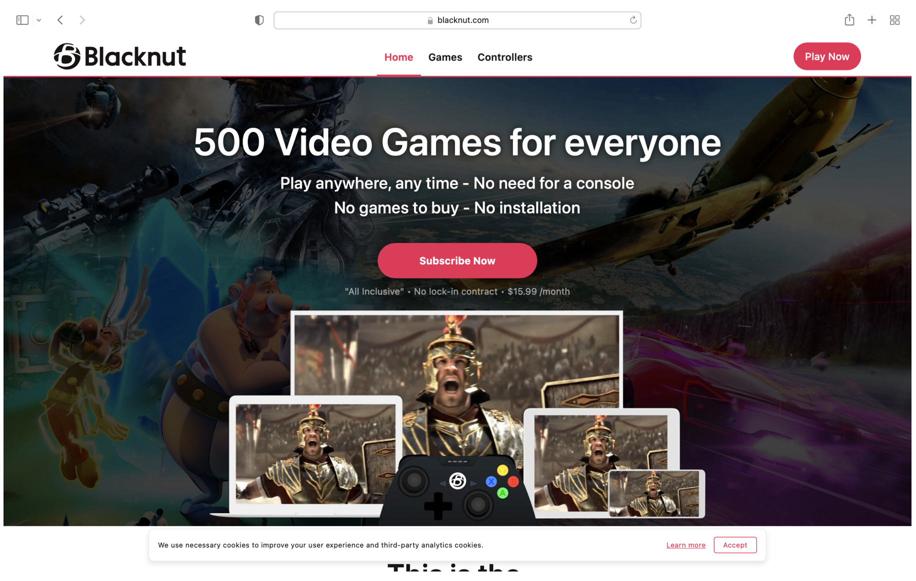

blacknut.com
Blacknut
Home
Games
Controllers
Play Now
500 Video Games for everyone
Play anywhere, any time - No need for a console
No games to buy - No installation
Subscribe Now
"All Inclusive" • No lock-in contract • $15.99 /month
We use necessary cookies to improve your user experience and third-party analytics cookies.
Learn more
Accept

Figure 4.1: Homepage.

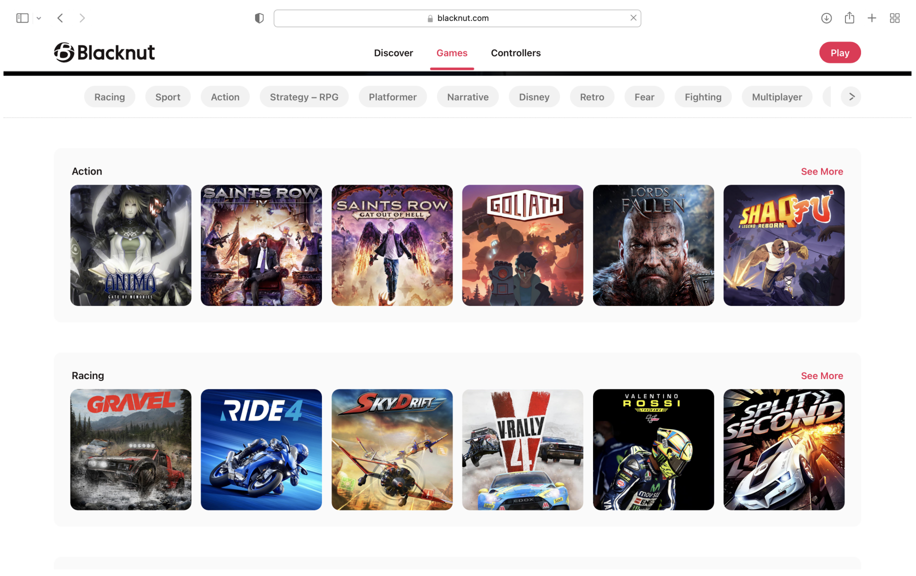
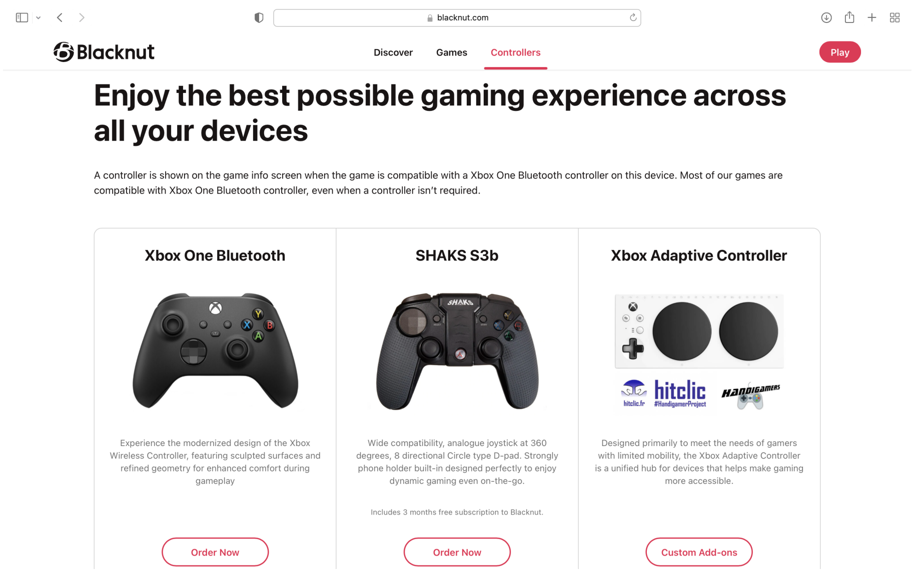


Figure 4.2: Catalogue.



Enjoy the best possible gaming experience across all your devices

A controller is shown on the game info screen when the game is compatible with a Xbox One Bluetooth controller on this device. Most of our games are compatible with Xbox One Bluetooth controller, even when a controller isn't required.

Xbox One Bluetooth

Experience the modernized design of the Xbox Wireless Controller, featuring sculpted surfaces and refined geometry for enhanced comfort during gameplay

Order Now

SHAKS S3b

Wide compatibility, analogue joystick at 360 degrees, 8 directional Circle type D-pad. Strongly phone holder built-in designed perfectly to enjoy dynamic gaming even on-the-go.

Includes 3 months free subscription to Blacknut.

Order Now

Xbox Adaptive Controller

Designed primarily to meet the needs of gamers with limited mobility, the Xbox Adaptive Controller is a unified hub for devices that helps make gaming more accessible.

Custom Add-ons

Figure 4.3: Controllers.

4.2.2. Setting up the A/B test

When there is the need to set up a new experiment, the researchers can specify the algorithms and configurations they want to test. The system randomly assigns each user present in the system to an algorithm, and the user sees the recommendations produced by that algorithm. The assignment of the algorithms is kept consistent for all duration of the experiment. When new users enter the system, they are randomly assigned to an algorithm, so this functionality redirects the traffic to a specific algorithm.

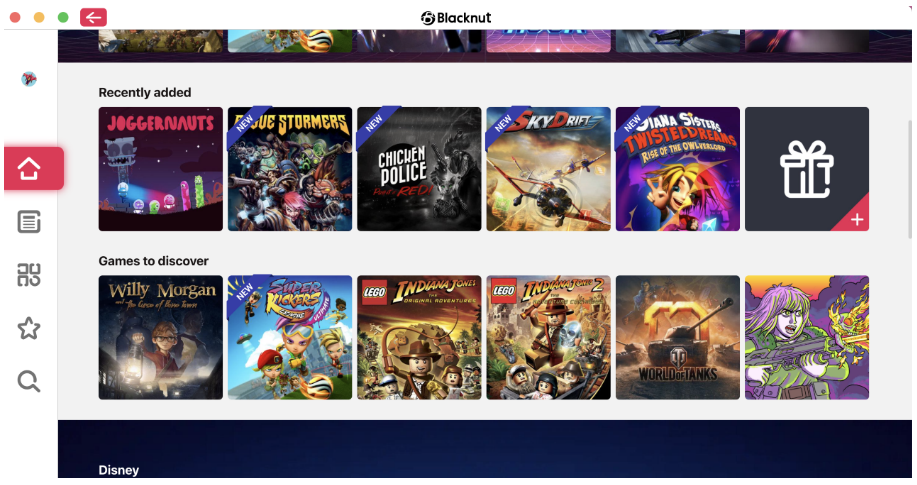


Figure 4.4: Games to discover.

4.2.3. User functionalities

Even though the user interface wasn't under our responsibility and Blacknut managed it, it's still significantly correlated with the Recommender System and with A/B testing, and it's important to have a perspective of what a real user can do. As in many real-life systems, the user is let free to browse the catalogue, as shown in Figure 4.2, to search for a particular game and, of course, to see possible recommendations. The list of recommended games is called "Game to Discover" Figure 4.4.

4.2.4. Collect user feedback

Any Recommender System must gather user-item interactions to work. As in many real applications, we don't have an explicit dataset composed of ratings; what we have are raw user click logs, which will be very important for our user study. In our case, those clicks are sessions of game play made by the users, and we will call the table of all recorded sessions Streams, Figure 4.1. In other words, when a user starts to play a game and eventually ends the session, a new entry on the dataset is made with the recorded session, Table 4.2. We will make an example to clarify the concept, mind that it's a table of a real company, so we cannot disclose all the details for privacy reasons. To get the user interactions, we had access to the table Streams through the service BigQuery[2] powered by Google.

[2]`https://cloud.google.com/bigquery`

id	user	created at	duration	game name
120	John	03-05-2022	400	Asphalt 6
121	Lucie	03-05-2022	540	Frozen
122	Bob	03-05-2022	1200	Tennis 2

Table 4.1: Streams.

id	user	created at	duration	game name
120	John	03-05-2022	400s	Asphalt 6
121	Lucie	03-05-2022	540s	Frozen
122	Bob	03-05-2022	1200s	Tennis 2
123	John	03-05-2022	600s	Tennis 2

Table 4.2: Streams after a session of gameplay.

But the all the working algorithms developed in our RS require a URM to compute the recommendations. The dataset is represented by a $n \times m$ matrix R (the URM). The element $r(i, j)$ in R is the rating expressed by user i on item j. In our case values for $r(i, j)$ are in the range 1 to 10. If the value of $r(i, j)$ is 0, it means that the user i has not expressed an opinion about the item j. To translate the Streams into a URM we can use a function for this purpose, we had available 5 different functions for doing so (the administrator of the system can decide which function to use):

- `AverageGameDuration()`: for each user finds the game that was played for the longest time by summing up the length of all the sessions a game was played and selecting the highest , that variable will be called MAX. After that, for each for each game played at least once by the user, the grade is calculated with the following formula $10 \cdot (x/\text{MAX})$ where x is the total time that a user played the game which is evaluated by again, summing up the session lengths in which the user played that game.
- `AverageGameSession()`: for each user finds the game that is played the most on average by computing the average duration session for each game the user has played and selecting the highest, that variable will be called MAX. After that, for each game the user has played at least once, the grade is computed with the following formula $10 \cdot (x/\text{MAX})$ where x is the average duration the user has played that game.
- `NumberOfSession()`: for each user finds the game played more times, the number

of times that is played is called MAX. After that, for each game played at least once, it calculates the rating with the following formula $10 \cdot (x/\text{MAX})$, where MAX is the number of times the game which is evaluated was played.

- `HybridDurationSession()`: this is a hybrid function, as the name suggests, that computes the grades using the `AverageGameSession()` and `AverageGameDuration()`. For each user, the grade for each game played at least once, is computed by simply summing up the grade generated by the previously mentioned functions and diving the results by 2.
- `HybridDurationNumber()`: this is a hybrid function, as the name suggests, that computes the grades using the `AverageGameSession()` and `NumberOfSession()`. For each user, the grade for each game played at least once, is computed by simply summing up the grade generated by the previously mentioned functions and diving the results by 2.

5 | Design and Execution of the Studies

What will be presented in this chapter is the core of our work. It is composed of two separate studies, the first system-centric study and the second a user-centric study. First, we will define the methodology used for the studies and their results.

5.1. System-centric

The first phase of the evaluation process is to select promising candidates to test in a live production setting. System-centric evaluation provides a cheap and fast to measure dependent values as a function of a recommendation algorithm based on historical data, to predict future performances and finally.

5.1.1. Methodology

In evaluating the performance of a recommender system, we need to partition the dataset into two different parts: the first part, referred to as the training set, is dedicated to the construction of the model, and the second part, referred to as the testing set, is used for testing the model [7].
In this work, we use 5-fold cross-validation, repeat each experiment several times to consider the average results. The K folds, and thus the training and testing sets, are built as follows: for each user u, we consider its profile as a vector $(j_k)_k$ containing the items rated by u, which order is obtained with a random shuffle. The cell (u, j_k) of the user-item matrix (with $r_{(u,j_k)}$ as rating) is mapped to the fold No. k mod K.

5.1.2. Dependent variables

To measure the quality of the algorithms, we adopted two families of metrics. **Error accuracy metrics** measure how close the prediction of the rating is near to the actual rating. **Classification accuracy metrics** instead measure the system's quality of pro-

ducing a list of relevant items for the user. We adopted more than one family of metrics first of all because, in the literature, they are the most widely used, also in works like ours that try to verify which metric is a good predictor for the actual performances. And finally, to compare the two approaches and decide which is more suitable for our domain. In this work, we only focused on accuracy metrics and didn't consider beyond-accuracy metrics.
With classification metrics, as a threshold for selecting relevant items, we chose the rating of 8 out of 10. For the length of the recommendation list, we chose 6, the same number of recommended items in the user interface.

5.2. Execution system-centric study

Algorithms

For this experiment we had the advantage to evaluate many algorithms at a low cost. The algorithms evaluated offline were the following:

- **Random**: selects random items from the catalogue.
- **Itemavg**: the score predicted is the average rating for this item, no matter the user.
- **matrix factorization**: many approaches are for decomposing the matrix are possible, we tried two factorizer, stochastic gradient descendent and SVD++. This approach as many hyperparameter to tune such as the regularization term and the number of features, this step will be omitted, the result table will contain only the version that we found having the best results overall.
- **item based k nearest neighbour**: simple k-Nearest Neighbours methods based on the similarity between items, we tried this approach for different values of K(K=5,10,15,20).
- **user based k nearest neighbour**: simple k-Nearest Neighbours methods based on the similarity between user, we tried this approach for different values of K (K = 5, 10, 15, 20).

Dataset

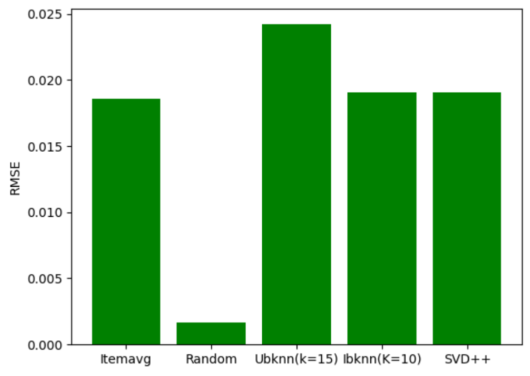
For the offline evaluation we used a data which contains browsing gaming logs(table Streams) of subscribers of Blacknut, on the Blacknut platform. The clicks were gathered in recommendation free scenario i.e. users could brows the catalogue but without seeing any form of recommendation. The dataset contains clicks, valid session of game play started ,from the 2021-10-01 to the 2022-03-01, on a total of 601 games. The Cleaned dataset comes from a polished version of the Original one, on which users with a short user profile are removed. We decided to remove players with a short user profile, that didn't interact a lot with the system, to make the dataset more suitable for our analysis and have more significant results. We removed players with less than 8 sessions launched.

	users	session	items
Original	85799	1261908	601
Cleaned	11521	958265	601

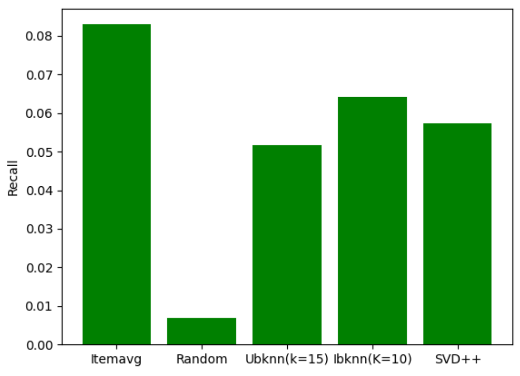
Table 5.1: Datasets for offline evaluation.

5.2.1. Results

Table 5.2 shows the result of the offline data-centric evaluation, performed with the methodology previously defined.
As expected, we see that matrix factorization performs better than classical memory-based approaches based on similarity metrics, while being evaluated with error accuracy metrics. Surprisingly, itemavg yields good results despite being a very simple method. Random, is the worst algorithm by far.
In contrast with the results obtained with error accuracy metrics, with classification accuracy metrics we obtained a different ranking of the algorithms. In this case memory-based approaches, in particular Ubknn (K = 15), outperformed the other techniques in terms of precision. Also in this family of metrics itemavg outperformed personalized algorithms, in this case the matrix factorization family, historically good in rating prediction task but less in top-n recommendation task. Again the random algorithm had very low performances, more than ten times lower than the other approaches.

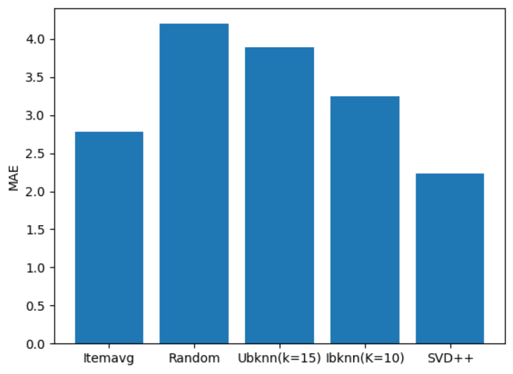


(a) Precision

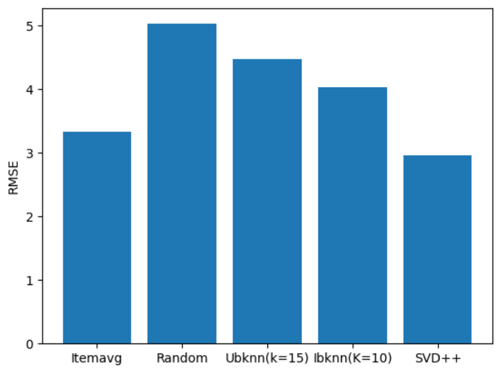


(b) Recall

Figure 5.2: Precision and Recall



(a) MAE



(b) RMSE

Figure 5.1: MAE and RMSE

Algorithm	RMSE	MAE	Precision@6	Recall@6
itemavg	3.332666	2.7867982	0.0186041	0.082981
random	5.03120423	4.198393	0.0016289	0.006845
Ubknn(K=5)	4,491834	3,889453	0.0220306	0.0576714
Ubknn(K=10)	4,480345	3,889453	0.0231306	0.0526012
Ubknn(K=15)	4,481847	3,889453	0.0242071	0.0516776
Ubknn(K=20)	4,616328	3,889453	0.0218610	0.0524798
ibknn(K=5)	4.1717397	3.250826	0.0193229	0.066696
ibknn(K=10)	4.036845	3.250826	0.0190656	0.064250
ibknn(K=15)	4.057676	3.250826	0.0183627	0.060656
ibknn(K=20)	4.054671	3.250826	0.0183015	0.060248
SGD	2.967335	2.3054353	0.013425	0.0595010
SVD++	3.0504135	2.2296327	0.014550	0.0572450

Table 5.2: Result offline evaluation.

5.3. User-centric

The goal of this study is to analyze the impact of a recommendation algorithm on real users of Blacknut by monitoring their behaviour, trough analysing system logs.

5.3.1. Procedure

For doing so, we aimed to perform an A/B test. Using the functionality implemented in the system, the researchers can specify the variants of the recommendation algorithm that they would like to test. The test participants are all the subscribers of Blacknut, i.e. the experimental units. All the participants gathered are then randomly divided into groups, and each group represents a particular version of a recommendation algorithm, an experimental condition. The participants are then kept consistent with the assignment of the recommendation algorithm for the all duration of the experiment, to experience the recommendations generated by the version they were assigned to. In our specif case, participants would enter the system and be exposed to the recommendations of an algorithm without knowing to which algorithm they were set. It's important to underline that participants could continue to browse the catalogue and were not restricted to choosing only from the recommendation list.
It's essential to agree upon an overall *evaluation criterion* to have some dependent variables to compare the performance of the algorithms and to state if one algorithm outperforms another. By reviewing the literature on this field [5, 19] and talking to the employees of Blacknut, more experts on what can be a good indicator that the game recommended was a good recommendation. The final answer was that the best way to measure that

is by looking at the revenues and whether the participants renew their subscription or not. Unfortunately, we could not determine this aspect directly. Thus we defined some objective metrics, objectively measurable attributes of the decision process and outcome, that are indirect measurements. Still, we believe they are coherent with our goal and make a good approximation of the recommendation effects, and they will be defined in section.
Regarding the duration of the experiment, it is not possible to decide that a priori. However, ideally, the experiment should stop when some statistically significant results are reached, not before, without damaging the company's interests because giving low-quality recommendations can be detrimental.

5.3.2. Participants

The participants of the experiments are Blacknut's subscribers paying the monthly subscription, unaware they are taking part in an experiment, and they are assigned to a different algorithm; this is done to remove the possibility of being biased. For privacy reasons, we do not have any information regarding the gender, age and country of the users. The only distinction between users that we made is about their seniority in the system; we divided the users between:

- **Old users**: users that took part in the experiment and already belonged to the system before the experiment started; in other words, they already had some user profile.
- **New users**: it is the second category that we took under consideration, participants that subscribed to Blacknut after the beginning of the experiment and still interacted with the recommender system and eventually built their user profile during the experiment.

5.3.3. Online metrics

We grouped the metrics in three categories based on relevance, novelty and user behaviour. The goal is to have some dependent variables to compare the performance of the algorithms. The metrics are objective values based on the information contained in the table Streams[1]

[1]The table containing all the interactions.

Relevance

Relevance is the ability of recommending items that very likely the user will appreciate. The following metrics aim to measure the relevance of a recommendation.:

- **Hit rate**

  It measures the ratio between the number of session started from a recommendation over the total number of session launched. It is defined as follow:

  $$\text{hit rate} = \frac{1}{|D|} \sum_{\forall i \in D} \frac{g_i}{s_i} \tag{5.1}$$

  where i belongs to the set of days of the experiment D. g_i is the number of session started from a recommendation during the day i of the experiment and s_i is the total number of session launched during the day i of the experiment.

- **Click trough rate**

  It is the ratio of games launched from a recommendation over the number of games recommended that day. It is defined as follow:

  $$\text{ctr} = \sum_{\forall i \in D} \sum_{\forall u \in U} \frac{t_{i,u}}{|R|} \tag{5.2}$$

  where i belongs to the set of days of the experiment D and u belong to U the set of players connected that day. $t_{i,u}$ is the number of games launched by player u in day i that were recommended and finally R in the set of games recommended that day.

- **Long ctr**

  It's a variant of the previous metric but in this case it measure not only if a games was played in the day i but also in the following n days.

  $$\text{long ctr} = \sum_{\forall i \in D} \sum_{\forall u \in U} \frac{k_{i,u}}{R|} \tag{5.3}$$

- **Coming back**

  Again the idea is to measure not only if an item was clicked but also if the user liked the recommendation; if a user comes back to play on a different day, he likely enjoyed playing the recommended game. This metric is defined as the ratio of games that the user played the day of the recommendation and that he came back to play the following n days over the total number of hits on that day.

$$\text{coming back} = \sum_{\forall i \in D} \frac{c_i}{h_i} \tag{5.4}$$

where i belongs to the set of days of the experiment D, c_i is the number of session started from a recommendation in the day i that led the user playing again that game in the following n days s_i is the total number of hit of the day i.

Novelty

Novelty is the ability of recommending items unknown to the user. This a difficult to measure without asking directly to a user if a recommended item is actually unknown to him, therefore we made the assumption that if the user has never played a game before, he probably doesn't know that game. The following metrics aim to measure the novelty of the recommendation:

- **Unique hit** This is the ratio between the number of games launched after a recommendation of games recommended that the user has never played before over the number of the total number of games launched after a recommendation. It is defined as follows:

  $$\text{unique hit} = \sum_{\forall i \in D} \frac{l_i}{p_i} \tag{5.5}$$

  where i belongs to the set of days of the experiment D. l_i is the number of session started from a recommendation in the day i that the user has never played before p_i is the total number of hit of the day i.

- **Novelty 30** This is the ratio between the number of games launched after a recommendation of games recommended that doesn't belong to the top-30 most popular game over the number of the total number of games launched after a recommendation. It is defined as follows:

  $$\text{novelty-30} = \sum_{\forall i \in D} \frac{f_i}{p_i} \tag{5.6}$$

  where i belongs to the set of days of the experiment D. f_i is the number of session started from a recommendation in the day i that doesn't belong to the top-30 most popular games p_i is the total number of hit of the day i.

User behaviour

This group of metrics is not strictly on whether a user clicks or not on a recommendation, but it measures more the impact of the system on the user's behaviour. This allows us to compare different periods, including time frames free of recommendations. For example, if we want to measure the impact of deployment of the recommender systems, we could compute these metrics in two distinct time frames, one before the deployment and the second after the deployment. The following metrics aim to measure the impact of the Recommender System on the user behaviour:

- **Session length** It is the average duration of all the session launched by the users.

  $$\text{SessionLength} = \frac{\sum_{\forall s \in S} \text{duration}_s}{S} \tag{5.7}$$

  Where s belongs to set of session launched during the duration of the experiment and duration_s is the duration of the session launched.

- **Different games** It is the average number of different games played by the users during a specific time frame.

  $$\text{DifferentGames} = \frac{\sum_{\forall u \in U} d_u}{U} \tag{5.8}$$

  where U is the set of users participating at the experiment, and d_u is the number of different games played during the duration of the experiment.

- **Number of streams** It is the average number of different games played by the users during a specific time frame.

  $$\text{NumberOfStreams} = \frac{\sum_{\forall u \in U} n_u}{U} \tag{5.9}$$

  where U is the set of users participating at the experiment, and n_u is the number of streams launched by the user u during the duration of the experiment.

5.4. Execution system-centric study

User-centric evaluation allows us to compare the perceived quality of recommendations in the different experimental conditions. In the previous sections, we defined the framework used for the experiment. Now, we give the details of the actual experiment and its results.

Experimental conditions

The experiment took place from 2022-03-09 to 2022-05-19. In total of 8348 participants, 5892 were classified as old users; the rest of the participants, 2456 users, were classified as new users (Subsection 5.3.2) were gathered, and 62610 sessions[2] were launched during this experimental period. The algorithms selected for the user-centric evaluation were three, so it was an A/B/C test:

- random: a non-personalized algorithm that selects random items to recommend from the catalogue.
- itemavg: a non-personalized algorithm that recommends the most rated items in the catalogue.
- matrix factorization: a personalized algorithm, we chose the SVD++ version for this experiment.

In total, we had nine experimental conditions[3], each algorithm evaluated for old users, new users and global that contains both categories. The Table 5.3 below summarizes the number of users participating in each experimental condition.

	random-old	random-new	random-global	itemavg-old	itemavg-new	itemavg-global	SVD++-old	SVD++-new	SVD++-global
Users	1819	843	2662	2100	790	2890	1973	823	2796
Sessions	12113	8407	20520	14223	8012	22235	11942	7913	19855

Table 5.3: Experimental conditions.

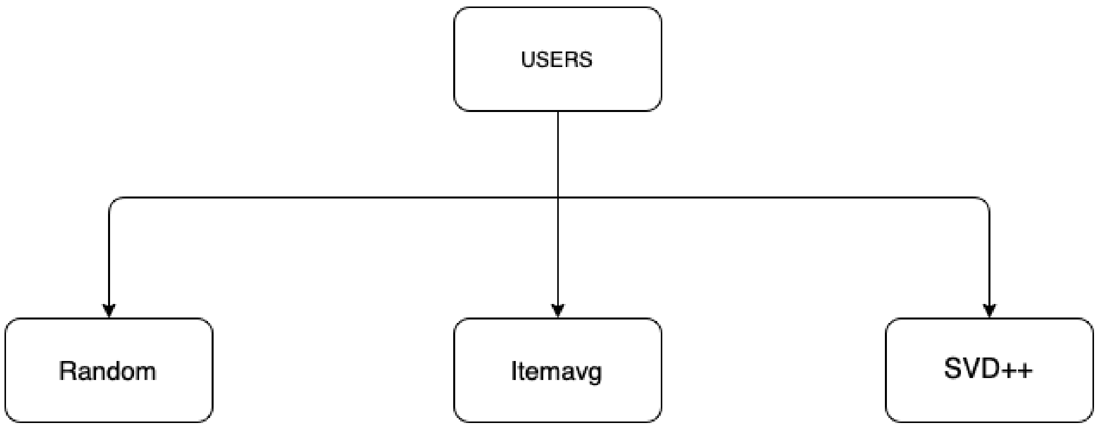


Figure 5.3: A/B/C test.

[2]A session is a game played by a user with its duration
[3]One of the possible conditions of the experiment

5.4.1. Results

We performed ANOVA over the nine experimental conditions, which depends on the type of algorithm and the seniority of the user in the system. ANOVA test confirmed that the results are statistically significant, returning a p-value < 0.05 on the measured dependent variables. From the metrics we previously defined, we now only give the results of the ones related to the relevance quality indicator. We limit the results to relevance because the metrics we defined for the Novelty section gave us trivial results for the algorithms we tested, such as random and itemavg. Also, the metrics regarding the user behaviour didn't lead to any significant results
The results show that for the objective metrics we measured (Figures 5.4, 5.5, 5.6, 5.7), we obtained an identical pattern. **Itemavg, a non-personalized algorithm that outperformed the matrix factorization algorithm for any group of users**. However, matrix factorization outperformed the random algorithms in any category; this shows that tailored recommendations still impact the user's choice. It's interesting to remark, but not surprisingly, that new users interacted more with the system than old users, who probably need fewer recommendations since they have a better knowledge of the catalogue. And the itemavg algorithm had a higher difference in performance in this category than the other categories (old users and global) compared to the matrix factorization, which needs a more extended user profile to perform better.

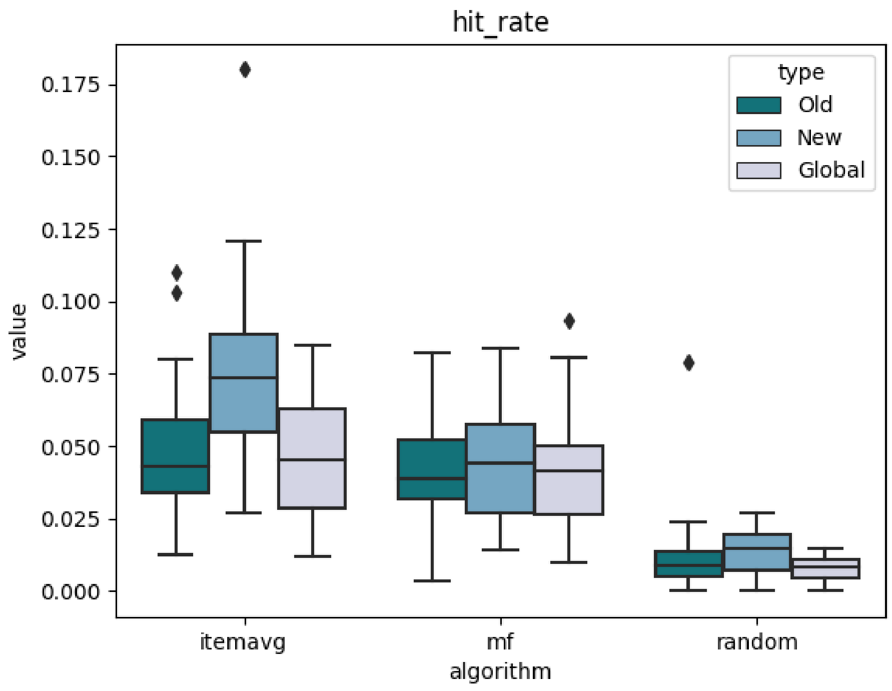


Figure 5.4: Hit-rate.

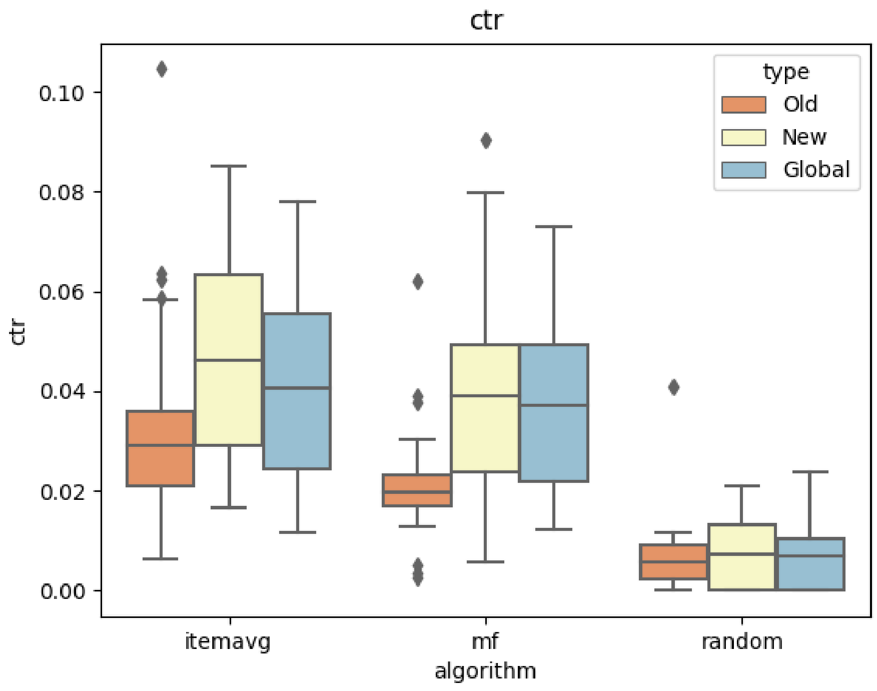


Figure 5.5: Click trough rate.

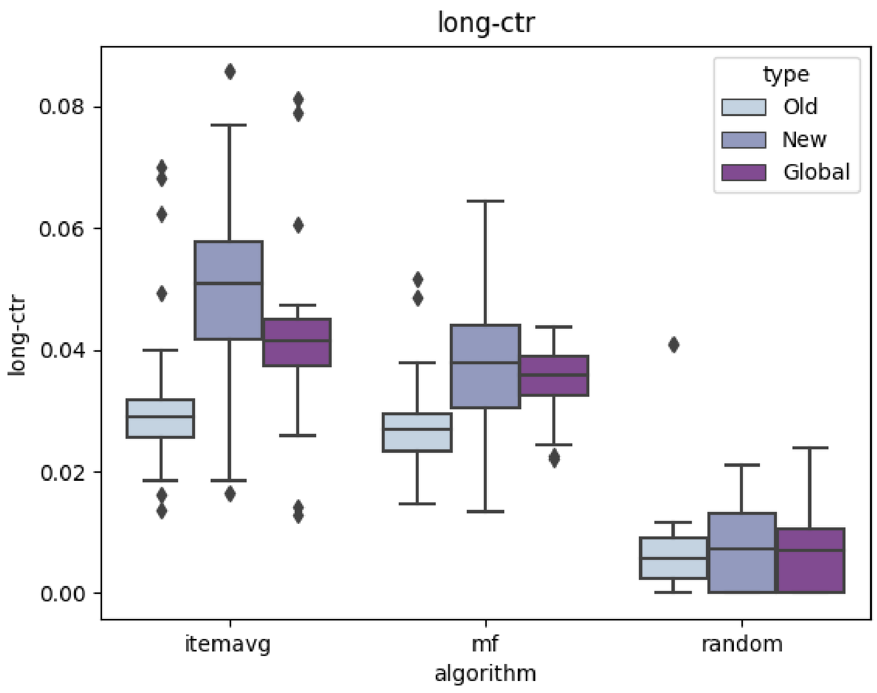


Figure 5.6: Long click trough rate (n=5).

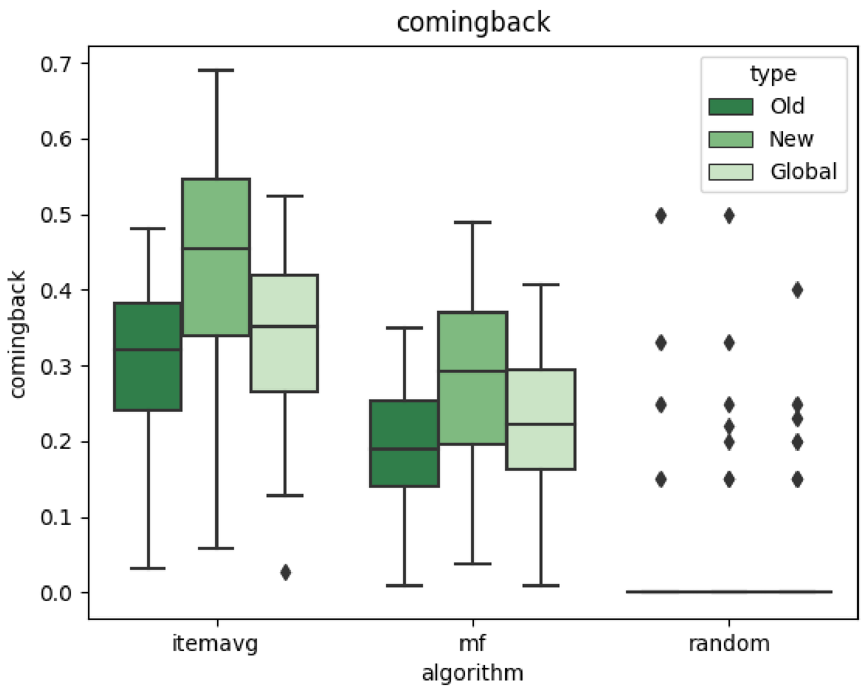


Figure 5.7: Comingback.

5.5. Limitations

Despite the statistical component that goes into their design and analysis, interpreting A/B tests remains partly art [16]. First, the measurement to approximate user engagement was designed by us to the best of our knowledge. Still, we couldn't find in the literature similar experiments in our same domain to compare with. Another possible limitation is the sample size; again, even if the experiment reached significant statistical results, we had a smaller sample size compared to other A/B tests in the Recommender Systems field [14, 16, 28]. Also, the catalogue size (500 items) can be considered undersized for a personalized algorithm. Finally, another significant limitation is that A/B test is strongly related to the user interface; we couldn't monitor where the user clicked but just if a game was launched.

6 | Conclusions and future developments

We have presented the experimental results of the system-centric and user-centric evaluation of the system in the different nine experimental conditions. The main research question in this study is: *In a video game platform, how is the online performance of the recommender systems related to the performance of such systems measured in terms of their accuracy?* Based on the evaluation outcomes of the two studies, we can claim that accuracy error metrics do not confirm the results obtained in the user-centric study. In fact, there are inconsistencies between the results obtained offline and the actual performance of the algorithms in the experiments. However, the previous result it's not surprising since it was already established that error metrics are not a good predictor in top-n recommendation task [4]. However, the ranking obtained during the system-centric evaluation with classification accuracy metrics respects the results obtained during the online phase. Hence the offline experiments indicate that precision and recall can indicate the quality of the recommendations and can be used as a good tool to select suitable candidates for deployment.

Despite the limitations, our work extends the dataset of studies regarding the evaluation of recommender systems. In particular, the relationship between system-centric and user-centric evaluation. In addition, according to our knowledge, we couldn't find any work on video game recommendations. Therefore this work can give hints on designing systems in this domain. Furthermore, the information gathered can be helpful in other environments with the right abstraction of the product's characteristics. Possible development of this project can be not to limit ourselves to accuracy metrics but to try out metrics that go beyond accuracy, as the current research trend in this field is trying to do [20].

Bibliography

[1] Analysis of Variance (ANOVA). In *Encyclopedia of Measurement and Statistics*. Sage Publications, Inc., 2455 Teller Road, Thousand Oaks California 91320 United States of America, 2007. ISBN 978-1-4129-1611-0 978-1-4129-5264-4. doi: 10.4135/9781412952644.n19. URL `https://methods.sagepub.com/reference/encyclopedia-of-measurement-and-statistics/n19.xml`.

[2] J. Beel, B. Gipp, S. Langer, and C. Breitinger. Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338, Nov. 2016. ISSN 1432-1300. doi: 10.1007/s00799-015-0156-0. URL `https://doi.org/10.1007/s00799-015-0156-0`.

[3] J. Bennett and S. Lanning. The Netflix Prize. *undefined*, 2007. URL `https://www.semanticscholar.org/paper/The-Netflix-Prize-Bennett-Lanning/31af4b8793e93fd35e89569ccd663ae8777f0072`.

[4] E. Campochiaro, R. Casatta, P. Cremonesi, and R. Turrin. Do Metrics Make Recommender Algorithms? In *2009 International Conference on Advanced Information Networking and Applications Workshops*, pages 648–653, May 2009. doi: 10.1109/WAINA.2009.127.

[5] H.-H. Chen, C.-A. Chung, H.-C. Huang, and W. Tsui. Common Pitfalls in Training and Evaluating Recommender Systems. *ACM SIGKDD Explorations Newsletter*, 19(1):37–45, Sept. 2017. ISSN 1931-0145. doi: 10.1145/3137597.3137601. URL `https://doi.org/10.1145/3137597.3137601`.

[6] M. Chen and P. Liu. Performance Evaluation of Recommender Systems. 2017. doi: 10.23940/IJPE.17.08.P7.12461256.

[7] P. Cremonesi, R. Turrin, E. Lentini, and M. Matteucci. An Evaluation Methodology for Collaborative Recommender Systems. In *2008 International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*, pages 224–231, Nov. 2008. doi: 10.1109/AXMEDIS.2008.13.

[8] P. Cremonesi, F. Garzotto, and R. Turrin. User-Centric vs. System-Centric Evalua-

tion of Recommender Systems. In P. Kotzé, G. Marsden, G. Lindgaard, J. Wesson, and M. Winckler, editors, *Human-Computer Interaction – INTERACT 2013*, Lecture Notes in Computer Science, pages 334–351, Berlin, Heidelberg, 2013. Springer. ISBN 978-3-642-40477-1. doi: 10.1007/978-3-642-40477-1_21.

[9] M. Dias, D. Locher, M. Li, W. El-Deredy, and P. Lisboa. The value of personalised recommender systems to e-business. pages 291–294, Jan. 2008. doi: 10.1145/1454008.1454054.

[10] M. Esteller-Cucala, V. Fernandez, and D. Villuendas. Evaluating Personalization: The AB Testing Pitfalls Companies Might Not Be Aware of—A Spotlight on the Automotive Sector Websites. *Frontiers in Artificial Intelligence*, 3:20, 2020. ISSN 2624-8212. doi: 10.3389/frai.2020.00020. URL `https://www.frontiersin.org/article/10.3389/frai.2020.00020`.

[11] A. Fabijan, P. Dmitriev, H. Holmstrom Olsson, and J. Bosch. Online Controlled Experimentation at Scale: An Empirical Survey on the Current State of A/B Testing. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 68–72, Aug. 2018. doi: 10.1109/SEAA.2018.00021.

[12] S. Fazeli, H. Drachsler, M. Bitter-Rijpkema, F. Brouns, W. v. d. Vegt, and P. B. Sloep. User-Centric Evaluation of Recommender Systems in Social Learning Platforms: Accuracy is Just the Tip of the Iceberg. *IEEE Transactions on Learning Technologies*, 11(3):294–306, July 2018. ISSN 1939-1382. doi: 10.1109/TLT.2017.2732349. Conference Name: IEEE Transactions on Learning Technologies.

[13] F. Garcin, C. Dimitrakakis, and B. Faltings. Personalized News Recommendation with Context Trees. *Proceedings of the 7th ACM conference on Recommender systems*, pages 105–112, Oct. 2013. doi: 10.1145/2507157.2507166. URL `http://arxiv.org/abs/1303.0665`. arXiv: 1303.0665.

[14] F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, and A. Huber. Offline and online evaluation of news recommender systems at swissinfo.ch. In *Proceedings of the 8th ACM Conference on Recommender systems*, RecSys '14, pages 169–176, New York, NY, USA, Oct. 2014. Association for Computing Machinery. ISBN 978-1-4503-2668-1. doi: 10.1145/2645710.2645745. URL `https://doi.org/10.1145/2645710.2645745`.

[15] A. Gilotte, C. Calauzènes, T. Nedelec, A. Abraham, and S. Dollé. Offline A/B Testing for Recommender Systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 198–206, New York, NY,

USA, Feb. 2018. Association for Computing Machinery. ISBN 978-1-4503-5581-0. doi: 10.1145/3159652.3159687. URL https://doi.org/10.1145/3159652.3159687.

[16] C. A. Gomez-Uribe and N. Hunt. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems*, 6(4):13:1–13:19, Dec. 2016. ISSN 2158-656X. doi: 10.1145/2843948. URL https://doi.org/10.1145/2843948.

[17] L. A. Hassanieh, C. A. Jaoudeh, J. B. Abdo, and J. Demerjian. Similarity measures for collaborative filtering recommender systems. In *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*, pages 1–5, Apr. 2018. doi: 10.1109/MENACOMM.2018.8371003.

[18] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, Jan. 2004. ISSN 1046-8188. doi: 10.1145/963770.963772. URL https://doi.org/10.1145/963770.963772.

[19] D. Jannach and M. Jugovac. Measuring the Business Value of Recommender Systems. *ACM Transactions on Management Information Systems*, 10(4):16:1–16:23, Dec. 2019. ISSN 2158-656X. doi: 10.1145/3370082. URL https://doi.org/10.1145/3370082.

[20] D. Jannach, P. Pu, F. Ricci, and M. Zanker. Recommender Systems: Past, Present, Future. *AI Magazine*, 42(3):3–6, Nov. 2021. ISSN 2371-9621. doi: 10.1609/aimag.v42i3.18139. URL https://ojs.aaai.org/index.php/aimagazine/article/view/18139. Number: 3.

[21] M. Kaminskas and D. Bridge. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Transactions on Interactive Intelligent Systems*, 7(1):2:1–2:42, Dec. 2016. ISSN 2160-6455. doi: 10.1145/2926720. URL https://doi.org/10.1145/2926720.

[22] R. Kohavi and R. Longbotham. Online Controlled Experiments and A/B Testing. In C. Sammut and G. I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 922–929. Springer US, Boston, MA, 2017. ISBN 978-1-4899-7687-1. doi: 10.1007/978-1-4899-7687-1_891. URL https://doi.org/10.1007/978-1-4899-7687-1_891.

[23] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Dis-*

covery, 18(1):140–181, Feb. 2009. ISSN 1573-756X. doi: 10.1007/s10618-008-0114-1. URL `https://doi.org/10.1007/s10618-008-0114-1`.

[24] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, Aug. 2009. ISSN 1558-0814. doi: 10.1109/MC.2009.263. Conference Name: Computer.

[25] M. Ludewig and D. Jannach. User-centric evaluation of session-based recommendations for an automated radio station. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, pages 516–520, New York, NY, USA, Sept. 2019. Association for Computing Machinery. ISBN 978-1-4503-6243-6. doi: 10.1145/3298689.3347046. URL `https://doi.org/10.1145/3298689.3347046`.

[26] A. Maksai, F. Garcin, and B. Faltings. Predicting Online Performance of News Recommender Systems Through Richer Evaluation Metrics. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 179–186, New York, NY, USA, Sept. 2015. Association for Computing Machinery. ISBN 978-1-4503-3692-5. doi: 10.1145/2792838.2800184. URL `https://doi.org/10.1145/2792838.2800184`.

[27] B. Marlin, R. S. Zemel, S. Roweis, and M. Slaney. Collaborative Filtering and the Missing at Random Assumption. *arXiv:1206.5267 [cs, stat]*, June 2012. URL `http://arxiv.org/abs/1206.5267`. arXiv: 1206.5267.

[28] A. Mogenet, T. A. N. Pham, M. Kazama, and J. Kong. Predicting online performance of job recommender systems with offline evaluation. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, pages 477–480, New York, NY, USA, Sept. 2019. Association for Computing Machinery. ISBN 978-1-4503-6243-6. doi: 10.1145/3298689.3347032. URL `https://doi.org/10.1145/3298689.3347032`.

[29] X. Ning and G. Karypis. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *2011 IEEE 11th International Conference on Data Mining*, pages 497–506, Dec. 2011. doi: 10.1109/ICDM.2011.134. ISSN: 2374-8486.

[30] L. Peska and P. Vojtas. Off-line vs. On-line Evaluation of Recommender Systems in Small E-commerce. *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, pages 291–300, July 2020. doi: 10.1145/3372923.3404781. URL `http://arxiv.org/abs/1809.03186`. arXiv: 1809.03186.

[31] P. Pu, L. Chen, and R. Hu. A user-centric evaluation framework for recommender

systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 157–164, 2011.

[32] F. Ricci. Recommender Systems Handbook | SpringerLink. URL https://link.springer.com/book/10.1007/978-0-387-85820-3.

[33] M. Rossetti, F. Stella, and M. Zanker. Contrasting Offline and Online Results when Evaluating Recommendation Algorithms. pages 31–34, Sept. 2016. doi: 10.1145/2959100.2959176.

[34] A. Said and A. Bellogín. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender systems*, RecSys '14, pages 129–136, New York, NY, USA, Oct. 2014. Association for Computing Machinery. ISBN 978-1-4503-2668-1. doi: 10.1145/2645710.2645746. URL https://doi.org/10.1145/2645710.2645746.

[35] G. Shani and A. Gunawardana. Evaluating Recommendation Systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, Boston, MA, 2011. ISBN 978-0-387-85820-3. doi: 10.1007/978-0-387-85820-3_8. URL https://doi.org/10.1007/978-0-387-85820-3_8.

[36] P. B. Thorat, R. M. Goudar, and S. Barve. Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System. *International Journal of Computer Applications*, 110(4):31–36, Jan. 2015. URL https://www.ijcaonline.org/archives/volume110/number4/19308-0760. Publisher: Foundation of Computer Science (FCS).

[37] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences of the United States of America*, 107(10):4511–4515, Mar. 2010. ISSN 1091-6490. doi: 10.1073/pnas.1000488107.

List of Figures

List of Tables