ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

UNIDAD DE TITULACIÓN

DISEÑO DE UN SISTEMA DE RECOMENDACIÓN BASADO EN GANANCIAS QUE USA MACHINE LEARNING PARA BALANCEAR LOS BENEFICIOS PARA EL USUARIO Y LA EMPRESA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL GRADO DE MAGISTER EN SISTEMAS DE INFORMACIÓN

JUAN FERNANDO RIOFRÍO VALAREZO
juan.riofrio@epn.edu.ec

DIRECTORA: PhD. LORENA KATHERINE RECALDE CERDA
lorena.recalde@epn.edu.ec

CODIRECTORA: PhD. ROSA DEL CARMEN NAVARRETE RUEDA
rosa.navarrete@epn.edu.ec

Quito, diciembre del 2022

APROBACIÓN DEL DIRECTOR

Como director del trabajo de titulación DISEÑO DE UN SISTEMA DE RECOMENDACIÓN BASADO EN GANANCIAS QUE USA MACHINE LEARNING PARA BALANCEAR LOS BENEFICIOS PARA EL USUARIO Y LA EMPRESA desarrollado por Juan Fernando Riofrío Valarezo, estudiante de la Maestría en Sistemas de Información Mención Inteligencia de Negocios y Analítica de Datos Masivos, habiendo supervisado la realización de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la Defensa oral.

LORENA KATHERINE RECALDE CERDA

Firmado digitalmente por LORENA KATHERINE RECALDE CERDA
Fecha: 2022.12.30 10:36:00 -05'00'

PhD. Lorena Katherine Recalde Cerda
DIRECTORA DE PROYECTO

Firmado electrónicamente por:
ROSA DEL CARMEN NAVARRETE RUEDA

PhD. Rosa del Carmen Navarrete Rueda
CODIRECTORA DE PROYECTO

DECLARACIÓN DE AUTORÍA

Yo, Juan Fernando Riofrío Valarezo, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento. La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

JUAN FERNANDO RIOFRIO VALAREZO

Firmado digitalmente por JUAN FERNANDO RIOFRIO VALAREZO
Fecha: 2022.12.29 09:48:57 -05'00'

Juan Fernando Riofrío Valarezo

AGRADECIMIENTOS

Quiero agradecer a Lorena Recalde, sin su ayuda no hubiera podido lograr este trabajo. Gracias por introducirme a este tema de investigación tan apasionante y acompañarme en el proceso. Además Lorena es la mejor profesora que tuve en la maestría, me enseñó mucho en mi paso por la Escuela Politécnica Nacional.

Quiero agradecer a todos mis profesores desde los que me enseñaron en la escuela hasta los que me enseñaron en la maestría. Gracias al aporte de cada uno soy lo que soy ahora y todo logro que alcance desde ahora será gracias a cada uno de ustedes. Estoy muy agradecido con la educación pública del Ecuador, tuve acceso a las mejores instituciones educativas del Ecuador sin que el dinero fuera un impedimento para aprender y progresar.

A mi familia, especialmente a mis abuelos: Inés, Luis, Miguel y Nancy que siempre creyeron en mí. Gracias a ellos, a mis padres y mis hermanas soy la persona que soy ahora. Gracias a sus valores y conocimientos pasados a mí persona he podido avanzar en mi vida, intentando ser siempre mejor. Mi agradecimiento final se lo dejo para mi madre Greta Valarezo. Ella me ha ayudado tanto durante toda mi vida que sin mi madre no hubiera podido lograr ni de cerca lo que he logrado hasta ahora.

Juan Fernando Riofrío Valarezo

CONTENTS

Resumen 1

Abstract 2

1 INTRODUCTION 3

1.1 Research question . . . 6
1.2 General objective . . . 6
1.3 Specific objectives . . . 6
1.4 Scope . . . 6
1.5 Theoretical framework . . . 7
1.5.1 Recommender Systems . . . 7
1.5.2 ALS Matrix Factorization . . . 8
1.5.3 Profit-Aware Recommender Systems (PARS) . . . 9
1.5.4 Weighted Rank Aggregation . . . 10
1.6 Related work . . . 11

2 METHODOLOGY 14

2.1 Design Science Research (DSR) . . . 14
2.2 Proposed approach . . . 16
2.3 Data sets . . . 24
2.4 Metrics . . . 25
2.4.1 Accuracy . . . 25
2.4.2 Impact . . . 25
2.4.3 Normalized Discounted Cumulative Gain (NDCG) . . . 26
2.4.4 Precision . . . 27
2.4.5 Mean Absolute Error (MAE) . . . 27
2.4.6 Root Mean Squared Error (RMSE) . . . 27

3 RESULTS AND DISCUSSION 29

3.1 Results . . . 29
3.1.1 MovieLens-1M . . . 30
3.1.2 AmazonVG . . . 33

3.1.3 AmazonVG2 35
3.2 Research question 38
3.3 Discussion 39

4 CONCLUSIONS **41**

5 REFERENCES **42**

FIGURES INDEX

Figura 2.1 Design Science Research (DSR) methodology. 16
Figura 2.2 Accuracy vs Profit of MARS models for different T_R values 19
Figura 2.3 Accuracy vs Profit of MARS models for different weights of the profit attribute 20
Figura 2.4 Profit boxplot at Top20 of the standard, baseline and MARS methods . 21
Figura 2.5 Profit histogram at Top 20 of the standard, baseline and MARS methods 21

TABLES INDEX

Tabla 2.1 Description of the notation. 16
Tabla 2.2 Simple metrics of MARS method compared to the standard method . . 20
Tabla 2.3 General overview of standard method ranking, profit ranking and MARS method ranking for user u 22

Tabla 3.1 Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 0$ on MovieLens-1M data set 30
Tabla 3.2 Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 4.5$ on MovieLens-1M data set 31
Tabla 3.3 Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 5$ on MovieLens-1M data set 32
Tabla 3.4 Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 6$ on MovieLens-1M data set 33
Tabla 3.5 Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 0$ on AmazonVG data set 34
Tabla 3.6 Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 4.5$ on AmazonVG data set 35
Tabla 3.7 Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 4.5$, and taking into account the relevance of the items, the inverse of the price and the profit attributes on AmazonVG2 data set 36
Tabla 3.8 Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 4.5$, and taking into account the relevance of the items, the price and profit attributes on AmazonVG2 data set 37

RESUMEN

Este trabajo propone el diseño de un sistema de recomendación consciente de las ganancias denominado Sistema de Recomendación Multi-consciente (MARS, siglas en inglés), donde el impacto en el usuario está sujeto a ajustes. El proceso de re-ranqueo de artículos se basa en las ganancias generadas por el negocio y otros atributos importantes (el precio del artículo), y se realiza utilizando un método de agregación de ranking ponderada. Los pesos se optimizan iterativamente utilizando una variante del algoritmo de Gradiente Descendente, para entrenar el modelo y controlar el impacto deseado en el usuario. De esta forma, el modelo controla el impacto en el usuario para no comprometer la lealtad del cliente, al mismo tiempo que aumenta la rentabilidad para la empresa. Los pesos pueden ser únicos para cada usuario, lo que hace que este enfoque sea personalizado. El método MARS también controla el impacto en el usuario ajustando el límite de ranking T_R, que determina la calificación mínima que deben tener los elementos para volver a ranquearlos. Los experimentos mostraron que los modelos MARS tienen la capacidad de controlar el impacto en el usuario mientras aumentan la ganancia al variar los pesos asociados a los atributos considerados. Es importante notar que el entrenamiento del modelo se puede paralelizar; así, los pesos y resultados se pueden obtener para cada usuario por separado. Al implementar MARS en 3 conjuntos de datos diferentes y varias configuraciones, obtuvimos resultados prometedores en términos de equilibrio entre el impacto en el usuario y la ganancia generada.

PALABRAS CLAVE: Sistema de recomendación, Consciencia de las ganancias, Re-ranqueo, Ganancia, Impacto en el usuario

ABSTRACT

This work proposes the design of a profit-aware recommender system named Multi-Aware Recommender System (MARS), where the impact on the user is subject to adjustment. The item re-ranking process is based on the profit generated for the business and other important attributes (i.e. price of the item), and it is performed using a weighted rank aggregation method. The weights are optimized iteratively using a variant of the Gradient Descent algorithm, in order train the model and control the desired impact on the user. In this way, the model controls the impact on the user so as not to compromise the customer loyalty, while increasing the profit for the company. The weights can be unique for every user, making this approach personalized. The MARS method also controls the impact on the user by adjusting the ranking threshold T_R, which determines the minimum rating that items must have to be re-ranked. Experiments showed that MARS models have the ability to control the impact on the user while increasing the profit by varying the weights associated to the considered attributes. It is important to notice that the training of the model can be parallelized; thus, the weights and results may be obtained for each user separately. By implementing MARS on 3 different data sets and various configurations, we obtained promising results in terms of the balance between impact on the user and profit generated.

Keywords: Recommender System, Profit-Aware, Re-ranking, Profit, Impact on the user

1 INTRODUCTION

Recommender systems are widely used by a large number of companies worldwide to recommend products to their customers. Generally, what is sought with the recommendations made to the client is to benefit the company, suppliers or even the client, and it is commonly intended to satisfy several of these at the same time [1]. Recommender systems serve several purposes depending on who they want to generate value for and in what way. For example, from the user's point of view, the objective is to find products that meet their preferences in the short or long term, or to support the user in the decision-making process; while from provider's point of view, it may create additional demand or increase the engagement of users [2]. It is important to take into account that the benefit for the user is indirectly transformed into a benefit for the company [1], since the user loyalty is built. This makes the user want to buy more products and stay in a specific company generating profit for the business [3]. The main benefit of recommender systems for the customers is that they help them find products that the users like and that they might not find on their own, while the main benefit for the business is to sell more products due to the correct recommendations [4]. Given these benefits, the field of recommender systems research has been greatly exploited seeking to improve these tools and produce greater and better benefits for both the client and the company.

There are different techniques for creating recommender systems that have been used by researchers and companies. The main classification based on the algorithm used to create the recommender system proposes four different types [5]-[7]: content-based recommendations, collaborative recommendations, demographic recommendations, and hybrid approaches. There are other types, such as knowledge-based recommendations or community-based recommendations [7], which, although detailed in the literature, are not as exploited in the academic and business world. Among the types of recommender systems, is worth mentioning that the ones that have been studied the most are content-based recommender systems [8] and collaborative filtering recommender systems [9]. Even more studied are the

hybrid approaches, which are created by combining two or more different techniques in order to improve the recommendations presented to the user and overcome the drawbacks that a certain individual technique would present [10]. There are also different kinds of recommender systems depending on what they base their recommendations on. Most recommender systems are based on user tastes (items relevance), but there are also those that take other information into account. For example, those that base their recommendations on the profit (profit-aware [11]) generated to the company. There are recommender systems that base the recommendations on the information of the context (context-aware [12]) in which the recommendations are made to suggest products according to the circumstance. Recommender systems that take into account the information provided by time are called time-aware [13], which facilitates the monitoring of the evolution of user tastes and improves the recommendations. This work studies in detail Profit-Aware Recommender Systems (PARS).

Initially, recommender systems sought to make suggestions based on what the user was believed to like or need, but this approach does not always produce the greatest economic benefits for the company. Although, it is true that user satisfaction produces benefits for the business, other techniques have been applied to take into account the economic profit generated by each product sold and thus recommend products not only based on the items relevance, but also based on the profit generated by the products. The recommender systems that use this approach are PARS, since they take into account the probability that a user buys the product and also the profitability for the sellers [14]. This approach is not the best from the user perspective, but [15] have shown that a company can greatly benefit from providing users with recommendations that meet business needs. When taking into account the items profitability for the recommendations, we have to be careful thus the trust of the client in the company or in the recommendations is not compromised [16]. If customers stop buying from a certain company because they realize that the recommendations they are given are for expensive and unnecessary products, the PARS would be producing less profit and losing customers. When the trust of the user in the recommender system or the company is low, it is better to restore confidence with optimal recommendations for the user, even if this means reducing profits for a while [17]. As long as the seller makes recommendations that are similar to the customer's tastes, the customer maintains a high level of trust [16]. Then, knowing the impact that profit-based recommendations have on the client is of the utmost importance, since it directly influences the reputation of the company and the recommender system.

Given this scenario, it can be deduced that there is a certain degree of impact from the PARS towards users, as the optimal list of recommendations for the user is affected by introducing products that generate higher profits for the supplier. Current literature has proposed solutions including quantifying this impact and balancing the benefits for both parties. Considering the above, the novelty of this work lies in the attempt to establish a way to balance the benefits for both the user and the company using machine learning techniques in the in- and post-processing stages. Therefore, this work aims to control and balance the impact on customers and generate more profits for companies with the proposed recommender system. In other words, the objective is to produce recommendations to the users that are relevant for them, but also increase the profitability of the company by recommending products based on the profit they generate.

This work proposes the design of a PARS, where the impact on the user is subject of adjustment. The proposed recommender system considers different attributes like the preferences of the user, the price and the profit generated by each item. For each user, the items are ranked based on the highest predicted ratings. Then, the items with predicted ratings above a defined threshold are re-ranked. The re-ranking process is based on the profit generated for the business and other important attributes (i.e. price of the item), and it is performed using a weighted rank aggregation method. The weights used in the weighted rank aggregation method determine the importance given to each attribute (e.g. profit, user preference). The weights are optimized iteratively using a variant of the Gradient Descent algorithm, in order to train the model and control the desired impact on the user. The weights can be unique for every user, making this approach personalized. We performed the experiments over three data sets (described in Section 2.3), and measured the impact on the user for the new recommendations. The results show that the proposed method balances the impact on the user while generating higher profit for the business, and allows the service provider to customize this variable depending on the business needs.

The remaining of this paper is structured as follows: Section 1.5 presents the necessary theory concepts to understand this work. Section 1.6 extends the background of this work and explains similar works to the one proposed in this study. Section 2 describes the methodology, the metrics, data sets and overviews the proposed approach used in this study. Section 3.1 gathers the main results obtained from the experiments. Section 3.3 explores the significance of the work's results. Finally, the concluding remarks are drawn in Section 4.

1.1 RESEARCH QUESTION

This work aims to answer the following research question:

1. What are the parameters or algorithms that could help control the impact on the user introduced by Profit-Aware Recommender Systems when recommending the TopN items to a user?

1.2 GENERAL OBJECTIVE

Design a profit-based recommendation system using machine learning algorithms, in order to balance the benefits for both the user and the company.

1.3 SPECIFIC OBJECTIVES

- ❒ Carry out a literature review related to recommender systems and profit-aware recommender systems.
- ❒ Develop a theoretical framework that supports this research work.
- ❒ Collect data to be used in the next phase to train, test, and validate the recommender system.
- ❒ Design the recommender system, define its architecture, define the machine learning algorithm to use, and train the model.
- ❒ Evaluate the performance of the designed profit-aware recommender system in terms of the impact to the user.
- ❒ Present the results in the form of a high-level scientific paper.

1.4 SCOPE

The final goal of this work is to develop a PARS which controls the impact on the user introduced by profit-based recommendations. The design of the model is based on a published

work, but the limitations of the method presented in [11] are taken into account to design a method that controls the impact on the user. The impact on the user will be controlled through the implementation of parameters that favors the accuracy of the recommendations. At the same time, we will use machine learning algorithms to control the introduced impact on the user. We do not want to use several machine learning algorithms in the final implementation due to the computational cost that these kind of algorithms add to the method. Then, it is imperative to find the correct machine learning algorithm that can be used to control the impact on the user in our method.

The proposed method will be tested in 3 different data sets. The size of the data sets will be smaller than $(6000x8500)$ to avoid huge memory usage when working on the data sets. This data sets defined size is enough to train and test the proposed method. We will have to create some random variables to complete the desired attributes of the data sets. The implementation of the method is only going to be over the 3 data sets, no real-life implementation will be performed.

Even though, we are talking about creating a recommender system, we want to propose a method that is independent of the technique chosen to predict the missing ratings. We are putting all the effort of this research in creating a re-ranking method that uses the predictions of the recommender system to present a new TopN list of items to the users. We still depends on the accuracy of the recommender system predictions, but as long as the results are accurate, we do not care about the functioning of the used recommender system. We will use Matrix Factorization, a well-known technique, to predict the missing ratings.

1.5 THEORETICAL FRAMEWORK

1.5.1 Recommender Systems

Recommender systems have been used for years in online commerce businesses. In these businesses there is usually a large amount of information regarding products [18]. Making it difficult for the users to find what they would like among all the products. To address this problem, recommender systems make personalized product suggestions for a certain customer based on their information or tastes [19]. The benefits for the company when using recommender systems are increased sales, diversity in the sale of products, customer

loyalty, among others. While the benefits seen by the client are the ease of search, less search time, recommendations of articles of interest, among others, which translates into an increase in customer satisfaction and loyalty [3]. Given the great interest in recommender systems from the scientific community and many businesses that use them daily to increase their sales and profits, there is a lot of research on this topic. One of the lines of research that has made significant progress is the use of different machine learning algorithms in recommender systems; for example, alternating least squares (ALS) matrix factorization, among others.

1.5.2 ALS Matrix Factorization

Matrix factorization solves two main problems in recommender systems. First, it predicts the ratings for unrated items. Second, as stated by [20], matrix factorization is regularly used for database dimensionality reduction. The number of rows and columns of the predicted rating matrix are usually high, making it hard to store the information. The required storage is reduced by dividing the matrix R of size $U \times I$ into two smaller matrices P and Q of sizes $U \times K$ and $K \times I$, respectively. Where U is the number of rows of the original matrix (generally the number of users), I is the number of columns of the original matrix (generally the number of items), and K are the number of latent factors used in the model. The product of the two smaller matrices is equal to R^*, which is the approximation of R [21]:

$$R \approx R^* = PQ \tag{1.1}$$

Accordingly, to calculate the rating given by a user u to an item i is:

$$R^*_{ui} = P_u Q_i \tag{1.2}$$

where, P_u of size $1 \times K$ is the u^{th} row of P associated to user u; and Q_i of size $K \times 1$ is the i^{th} column of Q associated to item i. The problem to solve in matrix factorization is to calculate the matrices P and Q. To learn the factor vectors P_u or Q_i), matrix factorization minimizes the regularized squared error on the set of known ratings [22]:

$$\min_{P^*,Q^*} \sum_{(u,i)\in J} (R_{ui} - P_u Q_i)^2 + c(||P_u||^2 + ||Q_i||^2) \tag{1.3}$$

Where J is the set of pairs (u, i) for which the rating R_{ui} is known, and c is a constant that controls the extent of regularization. Because both P_u and Q_i are unknowns, the problem is non-convex. ALS matrix factorization solves this by fixing either P_u or Q_i and optimizing the other one. In this way, the non-convex problem becomes a quadratic problem with a globally optimal solution. ALS switches between keeping P_u fixed while optimizing Q_i, and keeping Q_i fixed while optimizing P_u. Then, the value of the objective function (Equation 1.3) monotonically decreases [23]. [24] proved that the ALS algorithm is easily parallelizable, as it can compute each P_u or Q_i independently from the others.

1.5.3 Profit-Aware Recommender Systems (PARS)

Within the classification of recommender systems, taking into account what they base on to make recommendations (awareness), we find PARS. PARS are models that not only consider the relevance of the products to make recommendations, but also take into account the profits generated by the different products, among other factors, when the final recommendations are presented [25]. PARS recommend the products that generates the most profits for the company taking less into account the tastes or information of the user, which introduces a bias to the list of recommendations. If the bias is noticeable and the recommendations do not meet the needs of the client, user trust can be compromised [26]. At this point, in the best case scenario, the user continues to make purchases without taking into account the recommendations of the seller; while in the worst case, it is possible that the client takes their business to another establishment. One of the goals and major challenges for PARS is to maximize long-term profits without compromising customer trust [27]. It is important for businesses to be careful when recommending items while taking into account the profit generated by this recommendations [16]. That said, profit-based recommender systems can be useful to increase profits of a company, but the impact it generates on the customer must be considered. Based on this problem, different solutions have been devised so as not to compromise the trust of the customer in the company or the recommender system, and at the same time increase the profitability of the company.

1.5.4 Weighted Rank Aggregation

Rank aggregation is a problem that has been extensively studied since the 18^{th} century. Rank aggregation methods generate a consensus ranking list from several individual lists of ranked items. This lists normally come from various sources and may represent the preference of the users, items properties, the service provider objectives, etc. Rank aggregation combine all these lists in a single consensus ranking list by applying different algorithms that reorganize the items according to the different scores obtained in each individual list [28]. This consensus ranking list sum up the input information of all the individual lists. Despite the fact that rank aggregation is an old problem, it still draws attention from researchers due to its importance in the fields of expert and intelligent systems, metasearch engines, information retrieval, multicriteria decision-making and social choice theory [29].

There are two main rank aggregation techniques, rank-based aggregation and score-based aggregation. Rank-based aggregation is the most commonly used approach, it uses the rank (position) obtained by the items in each individual list. We get the rank of the item based on a score or property that define whether an item should be ranked above or below another item. This individual ranked lists are combined to obtain a final ranking using the aggregated ranks to re-rank the items. Score-based aggregation uses the scores obtained by the items in each individual list instead of the rank of the item. Once the aggregation of the scores is completed the final ranking list is obtained by ranking the items based on the aggregated score [30].

[31] demonstrated the advantages of using weights when performing a rank aggregation task over a conventional one that uses ranks or scores alone. Using representative weights for each individual list, makes the approach customizable and we can give any individual list more or less relevance over the final ranking list. This work specifically uses a weighted score-based aggregation technique, which general functioning is represented in Equation 1.4

$$final\_score(i) = \sum_{l \in L} \lambda(l) * score_l(i) \tag{1.4}$$

where L is the set of individual lists being considered by the rank aggregation method, $score_l(i)$ is the score obtained by item i in the individual list l, $\lambda(l)$ is the weight associa-

ted to the individual list l, and $final\_score(i)$ is the aggregated score obtained by item i in the final list.

1.6 RELATED WORK

In this section, the related studies conducted in the field of PARS and rank aggregation recommender systems are presented. It is important to mention that the limitations in the model presented by [11] motivated this study. In this study, a PARS is designed to balance the accuracy and the profit of the items. The authors modified the re-ranking method presented by [32]. The designed PARS takes the recommendations made by a matrix factorization algorithm and re-ranks the items based on the profit generated to the company. The re-ranking step is performed for each user, and only the items surpassing a predicted rating threshold T_R are re-ranked to avoid accuracy reduction using the function in Equation 1.5:

$$rank_{profit}(i, T_R) = \begin{cases} rank_{profit}(i), & \text{if } R^*(u,i) \geq T_R \\ rank_{std}(i) + \alpha_u, & \text{if } R^*(u,i) \in [T_H, T_R) \end{cases} \tag{1.5}$$

where $rank_{profit}(i)$ is the funtion that ranks items according to the profit, $rank_{std}(i)$ is the function that ranks items according to the predicted rating, and α_u is the maximum value obtained by $rank_{profit}(i)$. Items with predicted ratings above T_R are ranked according to the profit, while items that are below T_R are ranked according to the predicted rating. Notice that all items that are below T_R get ranked behind of all items that are above T_R. This work showed that including the threshold T_R can control and balance the impact on the user while increasing the profitability, as only the items considered relevant (above T_R) are re-ranked. Although the impact on the user is considered, there is still a considerable impact on the user when recommending the topN items with this method. The final list presented to the user may have many new items introduced to it depending on the value of T_R. For example, we have X items with predicted rating over T_R, now all this X items are re-ranked based in the profit, and only the TopN items are presented to the user. In this method it does not matter if the item was in the X^{th} position according to the user preference, now can be presented 1^{st} if it has the best profit out of the X items. To avoid this great changes on the final list, the proposed method in this work uses rank aggregation to consider all the scores of the attributes, as well as a thereshold T_R to present the final TopN recommendation. The attributes are the

properties of the items on which the recommendations are going to be based (e.g. predicted rating, profit, price, etc.). Now, the ranking scores of the X items with predicted rating over T_R are added to obtain a final score. Thus, this method takes into account not only the profit, but also how high their predicted rating was. Another important modification is the use of weights in the proposed approach. The degree of importance given to each attribute is controlled by independent weights for each user. The proposed method is personalized for each user, avoiding compromising the trust of the user by making generalizations. In the method presented by [11], the algorithm only works for one atribute (i.e. profit), but refrains from considering other important attributes at the same time. The proposed method in this study, solves this problem with the weighted rank aggregation process, therefore, all the relevant attributes are added to get a final ranking score.

[33] proposed a multi-criteria collaborative filtering recommender system using learning to rank and rank aggregation. The approach is a three-step hybrid ranking order system that finds the topN list based on multiple criteria (attributes) of the items (i.e. overall rating, story, acting, direction and visuals of movies). The first step of the method decomposes the multi-criteria tensor of rank 3 into single-criteria user-item matrices. Each channel of the multi-criteria tensor of size ($Users \times Items \times NumberOfCriteria$) represents a matrix of size ($Users \times Items$) for a single criteria. The second step applies the list-wise FM learning-to-rank method [34] to each individual matrix to find the partial-ranked lists. The single-criteria matrices are rating matrices with missing values. By using learning-to-rank, the method finds the ranking of all the items for every user on every criteria. The third step uses rank aggregation to obtain a global-ranked list from the parital-ranked lists. This study used a rank-based method, specifically, the Borda Count method. Algorithm 1 summarizes the proposed method. The experiments over the Yahoo! Movie data set showed that the proposed approach is better for capturing the user preferences than traditional methods. The method also performs better in terms of recommendation accuracy.

Algorithm 1 Method proposed by [33]

Require: Multi-criteria tensor, Number of criteria
$SingleCriteriaMatrices \leftarrow MatrixDecomposition(MultiCriteriaTensor)$
for $k = 1$ **to** $NumberOfCriteria$ **do**
$PartialRankedLists[k] \leftarrow LearningToRank(SingleCriteriaMatrices[(k])$
end for
$FinalRankedList \leftarrow RankAggregation(PartialRankedLists)$
return $FinalRankedList$

The proposed method in this study differs from the method proposed by [33] mainly in three

points. First, [33] predicts the ranking of all the attributes, compared to the proposed method that only predicts the rating matrix and assumes the attributes matrices are complete (do not have missing values). In the proposed method, if an attribute matrix has missing values, we must first predict these values before passing the attribute matrix as input for the method. Second, [33] do not use weights to perform the rank aggregation process, in other words, they use rank aggregation rather than weighted rank aggregation. Third, [33] do not uses a ranking threshold like the one proposed by [11] to define what items can be re-ranked.

2 METHODOLOGY

2.1 DESIGN SCIENCE RESEARCH (DSR)

The field of recommender systems is widely studied by researchers and academics to develop models that present the best recommendations to customers and produce benefits for the company. A work related to recommender systems usually modifies existing techniques, proposes new methods to improve recommendations or seeks solutions to specific problems such as the cold start problem [35], among other problems such as those described in [36]. Typically, this process culminates in the delivery of a trained model that will predict how much a customer will like a product. Under this background, the chosen research methodology is Design Science Research (DSR). Once the literature has been reviewed and given the characteristics of this project, DSR stands out as the methodology to be used since this approach proposes an effective solution to the problem presented within this study. This paradigm presents a complete and well-defined conceptual framework for the process of designing and building solutions called artifacts (models, methods, instances, algorithms, etc.) [37], [38]. In the case of this study, the artifact would correspond to the recommendation model.

Due to the advantages and utilities that the DSR methodology presents, several studies concerning recommender systems have used this approach in their research. There are several recent studies that integrate both DSR and recommender systems; For example, in [39], they use such an approach to create a recommendation system designed to identify and recommend specific and achievable financial goals appropriate for each user depending on their financial situation. In [40], the authors follow the steps of DSR to create a model whereby domain experts (human intelligence) and recommender systems (artificial intelligence) work together for data monetization. Another work that uses DSR in the design of recommendation systems is [41], where they develop a platform for intelligent planning of user

meals based on their clinical conditions, based on recommendation systems and machine learning algorithms.

For the development of this work, the DSR research approach will be applied for the design of the recommendation model (artifact). The DSR paradigm is increasingly used in the fields of information systems and even in other areas, and is used to solve real-world problems by developing innovative solutions [42]. The DSR process consists of 6 activities in nominal sequence [43]:

1. Identification of the problem and motivation: in this phase the specific problem to be investigated will be defined and the value of a solution to said problem will be justified.

2. Definition of the objectives for a solution: in this phase, the objectives of a solution will be inferred based on the definition of the problem and the knowledge of what is possible. The objectives will be rationally inferred from the specification of the problem.

3. Design and development: in this phase the artifact will be created. In this activity, the desired functionality of the artifact and its architecture will be determined, and then the actual artifact will be created.

4. Demonstration: in this stage, the use of the created artifact to solve one or more instances of the problem will be demonstrated. This could involve its use in experimentation, simulation, case study, test, or other appropriate activity.

5. Evaluation: Phase in which you will observe and measure how well the artifact supports a solution to the problem. This activity will involve comparing the goals of the solution to the actual observed results of using the artifact in the demonstration step. At the end of this activity, it will be decided whether to return to activity 3 to try to improve the effectiveness of the artifact or to continue with the communication.

6. Communication: The problem and its importance, the artifact, its utility and novelty, the rigor of its design, and its effectiveness will be communicated to researchers and other relevant audiences such as practicing professionals. This activity will be carried out through the writing of a scientific paper.

Figure 2.1 shows the flowchart of the proposed methodology for this research work.

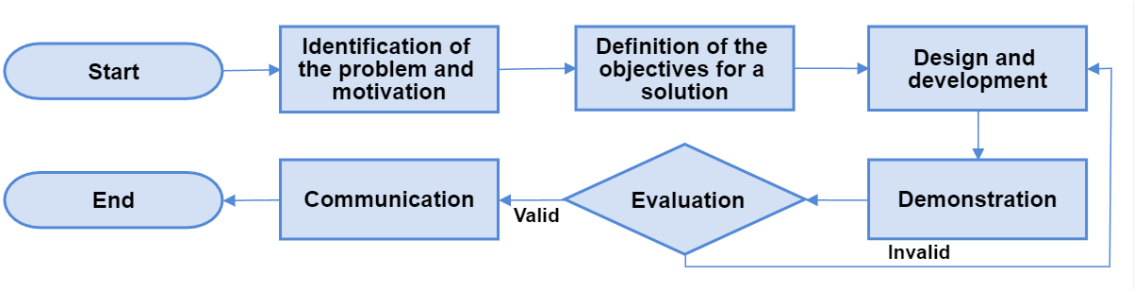


Figura 2.1: Design Science Research (DSR) methodology.

2.2 PROPOSED APPROACH

Notation	Description
U	Total users
I	Total items
R	Rating matrix
R^*	Predicted rating matrix(standard)
T_H	High Rating threshold
T_R	Ranking threshold
M	Total attribute matrices
V	Total attribute vectors
R_M	Rating attribute matrices
R_V	Rating attribute vectors
λ	Weight matrix
$rank_{std}$	Standard ranking/Ranking of R^*
$rank_M$	Ranking of attribute matrix
$rank_V$	Ranking of attribute vector
$rank_x$	Ranking of the proposed approach
α_u	Constant (sum of the highest rankings of each method)
γ	Learning rate
τ	Aggressive update factor

Table 2.1: Description of the notation.

Table 2.1 describes the notation used throughout this study. The proposed re-ranking method is independent from the algorithm used to predict the ratings. It is assumed that the predicted rating matrix R^* is the optimal on a user perspective. The method considers a ranking threshold T_R, which determines the items to re-rank for each user. T_R is a number between the minimum and maximum possible ratings. The re-ranking is only performed over items that were predicted above T_R to ensure an acceptable level of accuracy, and to reduce the impact on the users. T_R prevents items with low ratings from being presented to a certain user after re-ranking. The re-ranking is performed based on the ranking obtained by each item when considering other attributes, these attributes can be user-dependent or user-independent. If it is user-independent, only a vector of size I is considered. For example, the rating based

on the profit generated by each item is a user-independent attribute. Normally, the price and profit of items do not vary among users, all the users perceive the items equally according to this attribute. If the attribute is user-dependent, a matrix of size $U \times I$ is considered, meaning that every user rates each item differently. For example, the rating of every item based on the price preference of each user is a user-dependent attribute. This matrix can also be a predicted rating matrix from a different approach. For example, the predicted rating matrix resulting from applying a profit aware recommender system, or any other type of recommender system. When the attribute is user-independent a matrix can also be defined, but all the U rows are the same. It is important to mention that the higher ratings must be given to the best items while the lower values to the worst items for this method to work properly. As mentioned before, the values for the attributes can be considered as the rating, if and only if higher values are associated with the best items (i.e. profit), otherwise a conversion should be done. To control the extent to which an attribute is given importance, this method works with a weight matrix λ of size $U \times (1 + M + V)$. Each user have their own weights for every attribute, making this re-ranking method personalized for every user. The weight matrix can be updated according to the future usersínteractions with the recommendations. As this re-ranking method is able to consider several attributes to base the recommendations on, it is named MARS, which stands for Multi-Aware Recommender System.

The re-ranking method takes as inputs R^*, T_R, R_M, R_V, and λ. First, scaling is executed over R_M and R_V by following Equation 2.1.

$$R^*_K = \frac{R_K - \operatorname{mín} R_K}{\operatorname{máx} R_K - \operatorname{mín} R_K} * (\operatorname{máx} R^* - \operatorname{mín} R^*) + \operatorname{mín} R^* \tag{2.1}$$

where $K \in (M \cup V)$, are all the attribute matrices and vectors. This scaling ensures that all the rating attribute matrices and vectors are on the same scale $[\operatorname{mín} R^*, \operatorname{máx} R^*]$, avoiding very high, very low, or even negative values. Normally, values in R^* can be in range $(0, 5]$, but other positive values are also accepted. The next step is ranking the items based on the ratings obtained in the previous step. Ranking is performed by following Equation 2.2 over all the matrices and vectors, including R^*.

$$rank_J = (R^*_J + 1)^{-1} \tag{2.2}$$

where $J \in (K \cup std)$ are all the matrices and vectors(R^*, R_M, R_V) passed as inputs for the

ranking method. The operation $+1$ in Equation 2.2, ensures that the values obtained in the ranking step are in range $(0, 1]$, specifically in range $[1/(\max R^*+1), 1/(\min R^*+1)]$. Once all the ranking matrices and vectors are obtained, the re-ranking is performed. The re-ranking method only re-ranks items that have a rating $R^*(u, i)$ higher or equal to T_R, according to Equation 2.3. Otherwise, if the rating $R^*(u, i)$ is less than T_R, Equation 2.4 is used. Equation 2.5 describes the proposed method.

$$\lambda(u, std) * rank_{std}(u, i) + \sum_{m \in M} \lambda(u, m) * rank_m(u, i) + \sum_{v \in V} \lambda(u, v) * rank_v(i) \tag{2.3}$$

$$\lambda(u, std) * rank_{std}(u, i) + \alpha_u, \quad \text{where } \alpha_u = \sum_{j \in J} \max rank_j \tag{2.4}$$

$$rank_x(u, i, T_R) = \begin{cases} (2.3), & \text{if } R^*(u, i) \geq T_R \\ (2.4), & \text{else} \end{cases} \tag{2.5}$$

The constant α_u is calculated from the maximum rankings (worst rating) of each method. This constant ensures that items rated lower than T_R are not ranked ahead of items rated higher than T_R. Items with rating below T_R preserve their positions after re-ranking, while items with rating above T_R may change their position after re-ranking according to the attributes and their corresponding weights. Algorithm 2 shows how the proposed method works and how it executes.

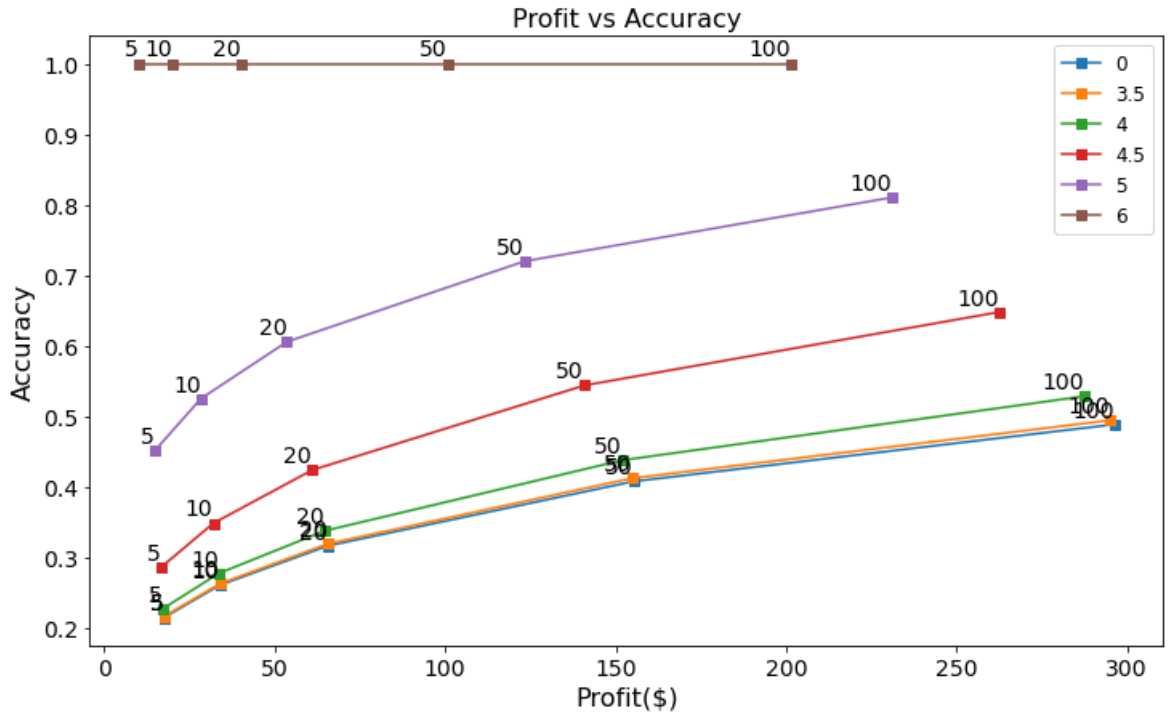
Algorithm 2 Proposed method

Require: R, R_M, R_V, λ, T_R
  $R^* \leftarrow RatingPredictionAlgorithm(R)$
  $R^*_M \leftarrow ScalingAlgorithm(R_M)$
  $R^*_V \leftarrow ScalingAlgorithm(R_V)$
  $rank_{std} \leftarrow (R^* + 1)^{-1}$
  $rank_M \leftarrow (R^*_M + 1)^{-1}$
  $rank_V \leftarrow (R^*_V + 1)^{-1}$
  $\alpha_u \leftarrow SumHighestRankings(rank_{std}, rank_M, rank_V)$
  if $R^* \geq T_R$ **then**
    $rank_x \leftarrow \lambda_{std} * rank_{std} + \lambda_M * rank_M + \lambda_V * rank_V$
  else
    $rank_x \leftarrow \lambda_{std} * rank_{std} + \alpha_u$
  end if
  return $rank_x$

The accuracy of the re-ranking method depends on the variables T_R and λ. Increasing the

T_R threshold towards $\max R^*$ results in higher accuracy, as fewer items are re-ranked based on the considered attributes, thus becoming more similar to $rank_{std}$. On the other hand, when decreasing T_R towards $\min R^*$, the method re-ranks more items (if $T_R = \min R^*$, all the items are re-ranked). At this point, the accuracy of $rank_x$ is completely controlled by λ. In both cases, if the weight given to the standard method is much greater than the weights of the attribute matrices and vectors ($\lambda(u, std) \gg \lambda(u, K)$), then, the re-ranked list is more similar to $rank_{std}$, resulting in high accuracy. On the other hand, decreasing $\lambda(u, std)$ or increasing $\lambda(u, K)$, results in a loss of accuracy because more importance is given to the attributes than to $rank_{std}$.

Figura 2.2: Accuracy vs Profit of MARS models for different T_R values



Therefore, choosing different values of T_R and different weights (λ) allows establishing the desired balance between accuracy and the objectives sought given the attributes. In particular, in the example of Profit-Awareness re-ranking illustrated in Figure 2.2, the accuracy in TopN recommendations of MARS could be improved by increasing T_R. At the same time, as shown in Figure 2.3, the accuracy in TopN recommendations could also be improved by decreasing the weight given to the *profit* attribute.

We performed an example of how MARS works on the MovieLens-1M data set at the Top20 recommendations. The parameters chosen for this experiment were $T_R = 4.5$, and ($\lambda(std) =$

Figura 2.3: Accuracy vs Profit of MARS models for different weights of the profit attribute

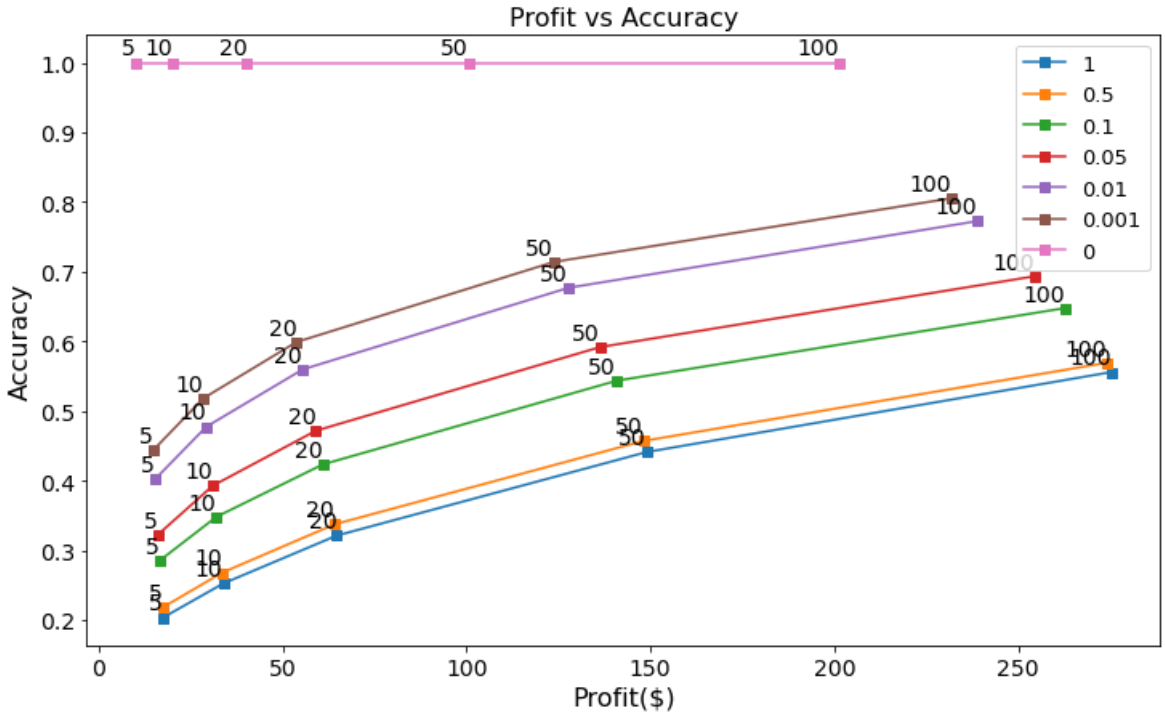


Table 2.2:
Simple metrics of MARS method compared to the standard method

	Accuracy	Impact	Profit	Items
Standard	1.0000	0.0000	40.31	2189
MARS	0.5596	0.4404	55.48	2106
Baseline	0.3058	0.6942	64.98	1839

$1, \lambda(profit) = 0.01]$). The method proposed by [11] was the baseline used to compare the performance of our model. In the experiment performed with this method, we only used the parameter $T_R = 4.5$, since it is not weighted. Table 2.2 shows the accuracy, the impact on the user introduced by each model, the average profit by user and the total number of different items being recommended. An accuracy of 55.96 % is accomplished with MARS. The impact perceived by the users is 44.04 %, meaning that out of 20 items, in average about nine new items are introduced to the list, compared to the standard. In other words, from the 20 items which represent the standard method list, nine items are exchanged for new items that will increase the profit for sellers. The baseline method obtained an accuracy of only 30.58 % and an impact on the user of 69.42 %, greater than the other methods. The profit obtained with MARS is $55.48, compared to $40.31 obtained by the standard and $64.98 obtained by the baseline method. The global diversity decreases in MARS and the baseline methods, as the items with better profit get recommended more, with the standard method

Figura 2.4: Profit boxplot at Top20 of the standard, baseline and MARS methods

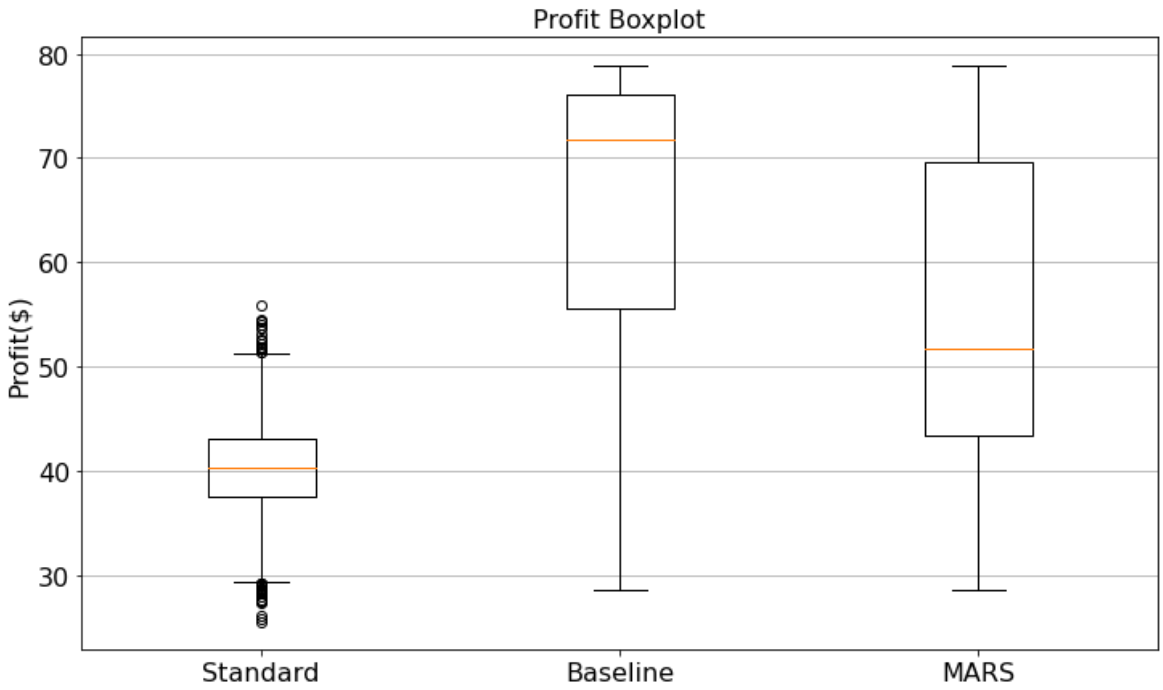
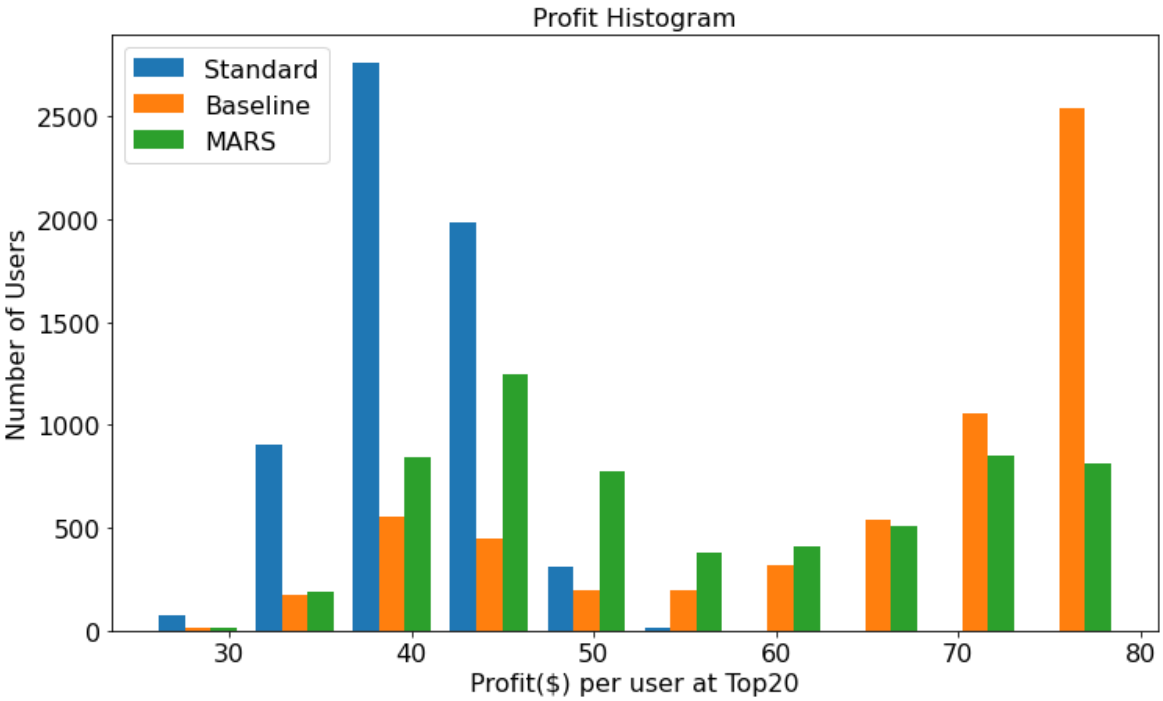


Figura 2.5: Profit histogram at Top 20 of the standard, baseline and MARS methods



2189 different items are recommended across users, while with MARS and the baseline only 2106 and 1839 different items are recommended respectively. The boxplot of the profit of all

the users for the standard, baseline and MARS methods is presented in Figure 2.4, while the histogram of the profit distribution in each method is shown in Figure 2.5.

Table 2.3:
General overview of standard method ranking, profit ranking and MARS method ranking for user u

Rating	Standard	Profit	MARS	Rating	Standard	Profit	MARS
5.00	1	1657	3	4.60	11	2887	16
5.00	2	2369	5	4.59	12	1922	12
4.94	3	1439	4	4.57	13	604	9
4.93	4	989	2	4.56	14	1872	13
4.85	5	535	1	4.53	15	245	7
4.82	6	1704	6	4.53	16	1226	10
4.77	7	3464	17	4.51	17	1732	14
4.75	8	2449	11	4.49	18	3004	18
4.65	9	934	8	4.48	19	1339	19
4.63	10	2499	15	4.48	20	1468	20

Table 2.3 shows the difference in the Top20 list for the standard method ranking, the profit ranking, and the MARS ranking. The rating column indicates the rating obtained for a certain item, the other columns show the position or ranking of a certain item in the corresponding method. The profit column is the item's position according to its profit. That is, the higher the profit an item generates, the lower (better) the position of the item on the ranking. For example, the item that was ranked 1^{st} in the standard method, in MARS it is ranked 3^{rd} because it is ranked 1657^{th} according to its profit, while the item ranked 5^{th} in the standard method, in MARS is ranked 1^{st} because it is ranked 535^{th} (the item has much better ranking) according to its profit. It is important to notice that $T_R = 4.5$, and that is the reason why the items with rating under 4.5 do not move in the MARS method ranking, even though they have a better profit ranking than other items.

As it has been shown in Figure 2.3, choosing different values of the weights leads to obtain the desired impact on the user. The optimization algorithm is explained in general in Algorithm 3. It is a variation of gradient descent. We want to find the optimal weights to obtain the desired impact. The cost function is computed using Equation 2.6:

$$cost@N = \frac{|Y - \hat{Y}|}{N} \tag{2.6}$$

where Y is the desired impact on the user and $\hat{Y}$ is the obtained impact on the user. The objective of the optimization step is to minimize this cost function. This means that the users should have an impact close to the desired one for the algorithm to converge. To solve this optimization process, we divided the update process of the weights in two parts. If the the

Algorithm 3 Weight optimization

Require: impact, N, iteration1,γ, iteration2, τ
  $\lambda \leftarrow initializeRandomWeights()$
  for $iteration = 1$ to $iteration1$ **do**
    $recommendations \leftarrow ProposedMethod(R, R_M, R_V, \lambda, T_R)$
    $impact^* \leftarrow getImpact(recommendations)$
    $cost \leftarrow getCost(impact, impact^*)$
    if $previousCost > cost$ **then**
      $\lambda \leftarrow weightsUpdate(\lambda, \gamma)$
    else
      break
    end if
  end for
  for $iteration = 1$ to $iteration2$ **do**
    $recommendations \leftarrow ProposedMethod(R, R_M, R_V, \lambda, T_R)$
    $impact^* \leftarrow getImpact(recommendations)$
    $cost \leftarrow getCost(impact, impact^*)$
    if $impact^* > impact$ **then**
      $\lambda \leftarrow aggresiveWeightsUpdate(\lambda, \tau)$
    end if
  end for
  return λ

error $Y - \hat{Y}$ is positive, then, there is not enough impact on the user and we can increase the impact. In this case, the weights related to the increase in impact are updated (e.g. $\lambda(profit)$). On the other hand, if the the error $Y - \hat{Y}$ is negative, it means that the impact on the user is higher than the desired and we should decrease the impact. In this case, the weights related to the decrease in the impact are updated (e.g. $\lambda(std)$). This update process follows Equation 2.7.

$$update(\lambda) = \begin{cases} \lambda(att) + \gamma(Y - \hat{Y}), & \text{if } Y - \hat{Y} > 0 \\ \lambda(std) - \gamma(Y - \hat{Y}), & \text{else} \end{cases} \tag{2.7}$$

where $\lambda(att)$ are the weights related to the attributes that increase the impact on the user, and $\lambda(std)$ are the weights related to the standard method and attributes that reduce the impact on the user. The learning rate γ indicates the size of the next step and controls the convergence process of the algorithm. The value of γ decays across iterations. The algorithm stops when some criteria are met, in this case, it stops if the cost function increases instead of decreasing after an iteration. It also stops if the cost functions difference is small, or if it reaches the specified number of iterations. This optimization problem finds a good solution, but may not find the optimal solution. Not all the users will have the desired impact.

Some users will have greater impact and others less impact. In this case, we want to update the users'weights that have a greater impact than desired. An aggressive update process is performed for this users by following Equation 2.8:

$$aggresiveUpdate(\lambda(std)) = \lambda(std) * \tau \tag{2.8}$$

In this case, the optimization algorithm only updates the weights of the attributes that decrease the impact on the user. The weights increase by a factor of τ, which increases with each iteration. This leads to the reduction of the impact on the user until the impact on all the users is lower or equal to the desired impact.

2.3 DATA SETS

The data sets used for the experiments are MovieLens-1M [44] and Amazon Video Game Review Data uploaded by [45]. For the MovieLens-1M data set, we created a random variable for the profit of each item. The values were chosen randomly using a Gaussian distribution with a mean value of $2 and defined minimum and maximum profit ($0 and $4, respectively), like proposed by [11]. The data set contains 1000209 ratings of 3706 different items performed by 6040 users. The sparsity of the data set is 0.0447.

On the other hand, the Amazon Video Game data set contains 32270 ratings of 5643 different items performed by 8369 users, the sparsity of the data set is equal to 0.00068. For this data set, since a price variable already exists, we calculated the profit from the Amazon referral fee and price of the item plus the closing fee. On this data set some items had low prices, and it did not make sense to charge a fee (profit for Amazon, and the profit being considered in the experiments) greater or close to the price of the item (profit for the seller). For the items with price lower than $3 we added $3 to the actual price. Therefore, the price of 423 items was updated. We assume that all the products in the Amazon Video Game data set belong to the category "Video Games". For this category, Amazon charges 15 % referral fee and $1.80 closing fee [46]. The profit of the items in this data set can range between $2.25 to $151.80, with a mean of $8.29 and a standard deviation of $10.61. We will refer to this data set as AmazonVG throughout this work.

The third data set used to perform the experiments is the same Amazon Video Game data

set, but in this case the referral fee is a random number between 10 % and 20 %. The profit is calculated from the new referral fee and the $1.80 closing fee. This was done to test the proposed method with 2 different attributes, specifically profit and price attributes. In the AmazonVG data set these two attributes are directly related by the same factor, and using the two attributes did not contribute to the final results. The profit of the items in this data set can range between $2.13 to $199.22, with a mean of $8.30 and a standard deviation of $11.12. We will refer to this third data set as AmazonVG2 throughout this work. The final data sets used in this paper can be found in https://www.kaggle.com/datasets/juanriofrio/mars-method.

2.4 METRICS

2.4.1 Accuracy

In this paper, the accuracy of the proposed approach is measured by the comparison between the list of TopN recommended items in the standard method ($rank_{std}$) and the list of TopN recommended items in the proposed method. The accuracy at TopN items for user u is defined in Equation 2.9:

$$accuracy@N = \frac{|std_N(u) \cap L_N(u)|}{N} \tag{2.9}$$

where N is the number of items being recommended (the length of the lists), $std_N(u)$ is the TopN recommendation list for user u based on the standard method (R^*), $L_N(u)$ is the TopN recommendation list for user u in the proposed approach ($rank_x$), $|std_N(u) \cap L_N(u)|$ is the number of items present in both lists (number of correct items recommended in the new ranking). Consequently, the number of new items introduced to the list is defined by $N - |std_N(u) \cap L_N(u)|$.

2.4.2 Impact

In this study, we are aiming to reduce or control the impact on the user arising from the re-ranking task. Then, it is important to mention that the impact is directly related to the accuracy and it is defined in Equation 2.10:

$$impact@N = 1 - accuracy@N \tag{2.10}$$

2.4.3 Normalized Discounted Cumulative Gain (NDCG)

The NDCG is an evaluation metric proposed by [47] and commonly used when a ranking task is performed in the information retrieval field. As stated by the authors, the benefit of this metric is that NDCG combines document rank (position) and degree of relevance (rating); which makes this metric perfect for this study. We can calculate the NDCG for any permutation of a set of items with known relevance, by comparing the Discounted Cumulative Gain (DCG) of the proposed approach with the DCG of the standard approach. The parameter N determines how many items to consider in the ranked lists [48]. We can express the $DCG@N$ for user u by Equation 2.11.

$$DCG@N = \sum_{i \in L_N(u)} \frac{2^{R^*(u,i)} - 1}{\log_2(1 + \hat{p}(i))} \tag{2.11}$$

where $L_N(u)$ is the TopN recommendation list for user u in the proposed approach ($rank_x$), item i is an element of $L_N(u)$, $R^*(u,i)$ is the predicted rating (relevance) of item i, and $\hat{p}(i)$ is the position of item i in the list $L_N(u)$. To calculate the $NDCG@N$, we need to calculate the $DCG@N$ of the proposed approach and the $DCG@N$ of the ideal ranking (standard method). We call this value $iDCG@N$ for user u and it is calculated by following Equation 2.12, while the $NDCG@N$ is calculated by using Equation 2.13.

$$iDCG@N = \sum_{i \in std_N(u)} \frac{2^{R^*(u,i)} - 1}{\log_2(1 + p(i))} \tag{2.12}$$

$$NDCG@N = \frac{DCG@N}{iDCG@N} \tag{2.13}$$

where $std_N(u)$ is the TopN recommendation list for user u in the standard method or the ideal TopN list, item i is an element of $std_N(u)$, $R^*(u,i)$ is the predicted rating of item i, and $p(i)$ is the position of item i in the list $std_N(u)$.

2.4.4 Precision

This metric compares the amount of truly highly ranked items to the amount of truly ranked items. The amount of truly ranked items is equal to the number of items in $L_N(u)$ that has been actually rated by the user u $(R(u,i) > 0)$. The amount of truly highly ranked items is equal to the number of items in $L_N(u)$ that has been actually rated above T_H by the user u $(R(u,i) \geq T_H)$

$$trulyRated@N = \sum_{i \in L_N(u)} \begin{cases} 1, & \text{if } R(u,i) > 0 \\ 0, & \text{else} \end{cases} \tag{2.14}$$

$$trulyHighlyRated@N = \sum_{i \in L_N(u)} \begin{cases} 1, & \text{if } R(u,i) \geq T_H \\ 0, & \text{else} \end{cases} \tag{2.15}$$

$$precision@N = \frac{trulyHighlyRated@N}{trulyRated@N} \tag{2.16}$$

2.4.5 Mean Absolute Error (MAE)

This metric calculates the average positions that one item in list $L_N(u)$ is moved, compared to the optimal list std_N. This is a positional metric, which means that it calculates the value using only the new position and the optimal position, and not the scores.

$$MAE@N = \frac{1}{N} \sum_{i \in L_N(u)} |p(i) - \hat{p}(i)| \tag{2.17}$$

where $p(i)$ is the position of the item i in the standard method or the optimal position, and $\hat{p}(i)$ is the position of the item i in the list $L_N(u)$ or the position predicted by the proposed approach.

2.4.6 Root Mean Squared Error (RMSE)

Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means that the RMSE should be more useful when large errors are particularly undesirable. This is a positional metric, which means that it calculates the

value using only the new position and the optimal position, and not the scores.

$$RMSE@N = \sqrt{\frac{1}{N} \sum_{i \in L_N(u)} (p(i) - \hat{p}(i))^2} \tag{2.18}$$

where $p(i)$ is the position of the item i in the standard method or the optimal position, and $\hat{p}(i)$ is the position of the item i in the list $L_N(u)$ or the position predicted by the proposed approach.

3 RESULTS AND DISCUSSION

3.1 RESULTS

In this section we present the main results derived from the experiments performed over the three data sets. Each subsection explains how the experiments were performed and the interpretation of the results. The results obtained from matrix R^* are considered the standard method. We assume that the ranking obtained by using this matrix is optimal on terms user tastes or the articles relevance. Therefore, all the new TopN lists obtained from different models are compared to the TopN list obtained from ranking items by their rating in R^*. Accuracy, impact, NDCG, RMSE and MAE metrics depend directly on the results obtained from the matrix R^*. That is the reason why the standard method has the best possible metrics in every experiment. We considered the method proposed by [11] to be the baseline method. We differentiate the MARS models by their attributes weights ratio. “MARS($ratio_\lambda$)"means a MARS model with a ratio of attributes weights equal to $ratio_\lambda$. The attributes weights ratio is calculated by following Equation 3.1:

$$ratio_\lambda = \frac{\sum \lambda(std)}{\sum \lambda(profit)} \tag{3.1}$$

where $\lambda(profit)$ are the weights related to the attributes that increase the impact on the user, and $\lambda(std)$ are the weights related to the attributes that decrease the impact on the user. From Equation 3.1 we can interpret that higher values of $ratio_\lambda$ means higher accuracy and lower values of $ratio_\lambda$ means higher impact on the user. In all the experiments we used $N = 20$ to obtain the TopN lists. For the experiments on AmazonVG2 data set, we used a different notation instead of MARS($ratio_\lambda$) due to the fact that there were 3 different attributes. In this case the notation was MARS($std_\lambda, profit_\lambda, price_\lambda$), this notation allowed us to differentiate the average weights given to each attribute by all the users. This notation uses Equation 3.2 to find the average of the weights of each attribute.

$$att_\lambda = \frac{\sum \lambda(att)}{U} \tag{3.2}$$

where att_λ is the average of the weights of the considered attribute, $\lambda(att)$ are the weights associated to the attribute, and U is the total number of users. Specifically in this experiment, $att \in [std, profit, price]$.

3.1.1 MovieLens-1M

For the experiments performed on the MovieLens-1M data set, first we used matrix factorization to calculate R^*. The values in the predicted matrix are in the range $R^*(u,i) \in [0,5]$. We performed four different experiments with MovieLens-1M data set by modifying the value of T_R. The chosen values for the experiments were $T_R = 0$, $T_R = 4.5$, $T_R = 5$, and $T_R = 6$. In each experiment, we calculate the metrics for the standard method, the baseline method and various MARS models with different weight distributions. The attributes considered for the experiments with this data set are the relevance of the items and the profit generated by each item. The weight associated to the relevance of the items is λ_{std}. Higher values of λ_{std} contribute to reduce the impact on the user. On the other hand, the weight associated to the profit generated is λ_{profit}. Higher values of λ_{profit} leads to higher profit.

Table 3.1:
Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 0$ on MovieLens-1M data set

	Accuracy	Impact	NDCG	RMSE	MAE	Precision	Profit	Items
Standard	1.0000	0.0000	1.0000	0.0000	0.0000	0.9040	40.31	2189
MARS(749.6017)	1.0000	0.0000	0.9999	0.9775	0.6875	0.9040	40.31	2189
MARS(159.1225)	0.9462	0.0538	0.9982	2.2354	1.7007	0.9050	42.56	2210
MARS(58.3852)	0.8528	0.1472	0.9921	4.7666	3.6049	0.9036	46.25	2195
MARS(17.0681)	0.6791	0.3209	0.9729	15.2338	11.0732	0.8944	52.70	2062
MARS(4.2983)	0.4281	0.5719	0.9292	65.3827	47.9337	0.8743	61.68	1655
MARS(0.0)	0.0068	0.9932	0.4781	2004.4725	1705.5477	0.5093	78.93	20
Baseline	0.0068	0.9932	0.4781	2004.4725	1705.5477	0.5093	78.93	20

The results obtained from the experiment using $T_R = 0$ are presented in Table 3.1. When $T_R \leq \min R^*$, all the items are re-ranked based on the selected attributes and their corresponding weights. A greater impact is obtained since all the items are re-ranked and more items have the chance of being introduced to the new Top20 list. At the same time, when $T_R \leq \min R^*$ and $\lambda_{std} \approx 0$, we achieved the same results as if we recommended only based on the profit generated by the items. This is the case of the baseline model and the MARS(0.0) model in Table 3.1, which obtained the same results. This two models recom-

mend the 20 most profitable items to all the users, decreasing the diversity, increasing the impact on the user and increasing the profit. Actually, these models have the greatest impact on the user among all the experiments, but they also achieved the highest profit of all the experiments performed with this data set. The maximum possible profit is achieved with these models. They increased the maximum obtainable profit by 95.81 %. This models greatly increased the MAE, obtaining a value of 1705.5477. This means that in average an item moved 1705 positions when re-ranked, affecting the accuracy of the models considerably. When $\lambda_{profit} \approx 0$, MARS model behaves like the standard model. The standard model and the MARS(749.6017) model got the same results in terms of accuracy, impact, profit, and number of items being recommended. The standard method had the best accuracy, while the baseline method had the worst accuracy of all the models. The results proved that higher values of $ratio_\lambda$ lead to higher accuracy and lower profit, while lower values of $ratio_\lambda$ produce lower accuracy and higher profit. MARS models have the ability to control the impact on the user by varying $ratio_\lambda$. When $T_R = 0$, MARS models can introduce an impact on the user in the range from 0 to 0.9932 and can generate profit in the range from $40.31 to $78.93. In other words, MARS models can reproduce the results of the standard model and the baseline model, plus any result in the middle in terms of impact on the user and profit. This experiment clearly shows that impact and profit are directly correlated, while accuracy and profit are inversely correlated.

Table 3.2:
Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 4.5$ on MovieLens-1M data set

	Accuracy	Impact	NDCG	RMSE	MAE	Precision	Profit	Items
Standard	1.0000	0.0000	1.0000	0.0000	0.0000	0.9040	40.31	2189
MARS(6.6967)	1.0000	0.0000	0.9985	1.0637	0.7511	0.9040	40.31	2189
MARS(4.074)	0.9661	0.0339	0.9969	1.9095	1.4571	0.9052	41.70	2206
MARS(3.5671)	0.9086	0.0914	0.9925	3.4505	2.6577	0.9059	43.91	2231
MARS(2.5901)	0.7733	0.2267	0.9803	10.1749	7.5398	0.9057	48.45	2273
MARS(1.4969)	0.5909	0.4091	0.9578	29.6759	23.0266	0.8999	54.27	2230
MARS(0.0)	0.3058	0.6942	0.8847	224.5113	188.3836	0.8871	64.98	1839
Baseline	0.3058	0.6942	0.8847	224.5113	188.3836	0.8871	64.98	1839

Table 3.2 shows the results from the experiment using $T_R = 4.5$. When $\min R^* < T_R < \max R^*$, some of the items are re-ranked. If T_R gets closer to $\max R^*$, less items are available for re-ranking, while if T_R gets closer to $\min R^*$ more items are available. A lower impact on the user is achieved when T_R gets closer to $\max R^*$. The baseline and MARS(0.0) models again got the same results and they both have the greatest impact on the user in this experiment, however they also got the highest profit of all the models in this experiment. The maximum possible profit in this experiment is achieved with these two models. This

models increased the MAE, obtaining a value of 188.3836. This means that in average an item moved 188 positions when reranked, affecting the accuracy of the models but not as much as the previous experiment. When $\lambda_{profit} \approx 0$, MARS model behaves like the standard model. The standard model and the MARS(6.6967) model attained the same results in terms of accuracy, impact, profit, and number of items being recommended. The standard method had the best accuracy, while the baseline method had the worst accuracy of all the models similarly to the previous experiment. This experiment also proved that higher values of $ratio_\lambda$ lead to higher accuracy and lower profit, while lower values of $ratio_\lambda$ produce lower accuracy and higher profit. When $T_R = 4.5$, MARS models can introduce an impact on the user in the range from 0 to 0.6942 and can generate profit in the range from $40.31 to $64.98. Increasing T_R from 0 to 4.5 decreased the impact on the user by a maximum of 30.10 % when we compared the MARS(0.0) models of the two experiments. This experiment also proved that the impact and profit are directly correlated, while accuracy and profit are inversely correlated.

Table 3.3:
Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 5$ on MovieLens-1M data set

	Accuracy	Impact	NDCG	RMSE	MAE	Precision	Profit	Items
Standard	1.0000	0.0000	1.0	0.0000	0.0000	0.9040	40.31	2189
MARS(2.4217)	1.0000	0.0000	1.0	0.5396	0.3535	0.9040	40.31	2189
MARS(1.8134)	0.9896	0.0104	1.0	0.8167	0.5653	0.9036	40.68	2196
MARS(1.4044)	0.9655	0.0345	1.0	1.3814	1.0241	0.9031	41.49	2219
MARS(1.0856)	0.9075	0.0925	1.0	3.1615	2.4595	0.9023	43.24	2277
MARS(0.7841)	0.8239	0.1761	1.0	9.3877	7.2642	0.9015	45.88	2362
MARS(0.0)	0.6053	0.3947	1.0	63.0612	52.6788	0.9001	53.57	2149
Baseline	0.6053	0.3947	1.0	63.0612	52.6788	0.9001	53.57	2149

Table 3.3 shows the results from the experiment using $T_R = 5$. When $T_R = \max R^*$, only the items with the highest possible rank ($\max R^*$) are re-ranked. As mentioned before, if T_R gets closer to $\max R^*$, less items are available for re-ranking. In this case, only the items with predicted rating of 5 were re-ranked. Similar to the previous experiments, the baseline and MARS(0.0) models got the same results and they both have the highest impact on the user and profit in this experiment. When $ratio_\lambda$ is high enough, MARS model behaves like the standard model. In this experiment, MARS(2.4217) was the model that got the same results as the standard model in terms of accuracy, impact, profit, and number of items being recommended. This experiment reaffirms the idea that higher values of $ratio_\lambda$ lead to higher accuracy and lower profit, while lower values of $ratio_\lambda$ produce lower accuracy and higher profit. We found a correlation between the variables T_R and $ratio_\lambda$ when we compared the result of all the experiments. When T_R was closer to $\min R^*$, $ratio_\lambda$ needed higher values to

obtain the same results as the standard model (i.e. when $T_R = 0$, $ratio_\lambda = 749.6017$). On the other hand, when T_R was closer to $\max R^*$, $ratio_\lambda$ needed lower values to obtain the same results as the standard model (i.e. when $T_R = 5$, $ratio_\lambda = 2.4217$). If we use $T_R = 5$, MARS models can introduce an impact on the user in the range from 0 to 0.3947, and the models can generate profit in the range from $40.31 to $53.57. Increasing T_R from 0 to 5, decreased the impact on the user by a maximum of 60.26 % when we compared the highest impact of the two experiments. It also decreased the maximum obtainable profit by 32.98 % when we compared the maximum obtainable profit of the two experiments. In this experiment it is important to notice that the NDCG did not changed and it stayed at 1. This was due to the fact that even though items were reranked, all of them have the rating equal to 5. The NDCG was not affected because all the items had the same relevance, and in terms of NDCG all the results of the models were equally valid.

Table 3.4:
Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 6$ on MovieLens-1M data set

	Accuracy	Impact	NDCG	RMSE	MAE	Precision	Profit	Items
Standard	1.0	0.0	1.0	0.0	0.0	0.904	40.31	2189
MARS(1.0146)	1.0	0.0	1.0	0.0	0.0	0.904	40.31	2189
MARS(0.0)	1.0	0.0	1.0	0.0	0.0	0.904	40.31	2189
Baseline	1.0	0.0	1.0	0.0	0.0	0.904	40.31	2189

Finally, the results obtained from the experiment using $T_R = 6$ on the MovieLens-1M data set are presented in Table 3.4. When $T_R > \max R^*$, no items are re-ranked independently of the attributes being considered and their weights. There is no impact on the user since no items are re-ranked and the new Top20 list is the same as the one in the standard method. It does not matter the values of the weights of the MARS model, the result is always going to be the same. The same happens with the baseline model, where no items are re-ranked and the result is the same as the standard method.

3.1.2 AmazonVG

For the experiments performed on the AmazonVG data set, first we used matrix factorization to calculate R^*. This data set was particularly hard to predict due to the sparsity of the data set. The values in the predicted matrix are in the range $R^*(u, i) \in [0.5036, 5.736]$. Normally, in recommender systems, predicted rating values over 5 are rounded to 5. In this data set we kept the values as predicted so that the system differentiates between items with rating over 5. Then, a value of 5.5 is better than a value of 5.3 and they are not considered equal

anymore. We performed two different experiments with AmazonVG data set by modifying the value of T_R. The chosen values for the experiments were $T_R = 0$ and $T_R = 4.5$. In each experiment, we calculate the metrics for the standard method, the baseline method and various MARS models with different weight distributions. The attributes considered for the experiments with this data set are the relevance of the items and the profit generated by each item. Compared to the previous data set, AmazonVG has higher profits associated to the items. The difference among the profit of the items of this data set is also higher, and the standard deviation is $10.61.

Table 3.5:
Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 0$ on AmazonVG data set

	Accuracy	Impact	NDCG	RMSE	MAE	Precision	Profit	Items
Standard	1.0000	0.0000	1.0000	0.0000	0.0000	0.8947	225.55	313
MARS(820.4217)	1.0000	0.0000	0.9999	1.0546	0.7017	0.8947	225.55	313
MARS(203.5705)	0.9029	0.0971	0.9982	5.0480	3.3009	0.8987	275.56	317
MARS(120.3925)	0.8020	0.1980	0.9934	13.4806	7.8038	0.9146	345.64	316
MARS(64.0147)	0.6090	0.3910	0.9752	53.5069	30.0941	0.9143	526.44	316
MARS(33.7112)	0.3427	0.6573	0.9299	247.5940	144.3340	0.9576	904.77	285
MARS(0.0)	0.0023	0.9977	0.7255	2582.6104	2209.6102	0.8867	1999.35	20
Baseline	0.0023	0.9977	0.7255	2582.6104	2209.6102	0.8867	1999.35	20

The results obtained from the experiment using $T_R = 0$ are presented in Table 3.5. As mentioned before, when $T_R \leq \min R^*$, all the items are re-ranked based on the selected attributes and their corresponding weights. A greater impact is reached since all the items are re-ranked. The baseline model and the MARS(0.0) model in Table 3.5, obtained the same results. These models recommend the 20 most profitable items to all the users, decreasing the diversity, increasing the impact on the user and increasing the profit. Actually, these models have the greatest impact on the user among all the experiments, but they also achieved the highest profit of all the experiments performed with this data set. The maximum possible profit is achieved with these models. The maximum obtainable profit in this data set increased by 8.8643 times, while the increase obtained in the MovieLens-1M data set was 1.9581 times. The big difference between this values is due to the fact that the profit generated in AVG data set greatly vary among the items with a minimum profit of $2.13 to a maximum profit of $199.22. In this data set, when $\lambda_{profit} \approx 0$, MARS model also behaves like the standard model. The standard model and the MARS(820.4217) model got the same results in terms of accuracy, impact, profit, and number of items being recommended. As in all the experiments, the standard method had the best accuracy, while the baseline method had the

worst accuracy of all the models. In this experiment, it was also proved that MARS models have the ability to control the impact on the user by varying the $ratio_\lambda$. When $T_R = 0$, MARS models can introduce an impact on the user in the range from 0 to 0.9977 and can generate profit in the range from $225.55 to $1999.35. In other words, MARS models can reproduce the results of the standard model and the baseline model, plus any result in the middle in terms of impact on the user and profit.

Table 3.6:
Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 4.5$ on AmazonVG data set

	Accuracy	Impact	NDCG	RMSE	MAE	Precision	Profit	Items
Standard	1.0000	0.0000	1.0000	0.0000	0.0000	0.8947	225.55	313
MARS(11.4108)	1.0000	0.0000	0.9998	1.0349	0.6896	0.8947	225.55	313
MARS(7.5259)	0.9129	0.0871	0.9981	4.6191	3.0449	0.8987	269.74	325
MARS(7.0263)	0.8241	0.1759	0.9937	11.8718	6.9592	0.9268	329.80	338
MARS(5.9813)	0.6446	0.3554	0.9762	47.1502	26.8836	0.9135	489.70	371
MARS(4.4852)	0.4173	0.5827	0.9364	189.8059	113.5658	0.9391	782.70	416
MARS(0.0)	0.1464	0.8536	0.7899	1464.8978	1265.5465	0.8926	1456.91	473
Baseline	0.1464	0.8536	0.7899	1464.8978	1265.5465	0.8926	1456.91	473

We consider the experiment using $T_R = 4.5$ on AVG data set to be the experiment closest to a real scenario. [11] showed that $T_R = 4.5$ was the optimal threshold using a simple model of relevance-based purchasing. If the rating of the items gets lower, the probability of purchase decays exponentially. The only change to the actual data set was the price of 423 items. Profit was calculated using actual Amazon referral and closing fees. Table 3.6 shows the results of the experiment under this conditions. The results of this experiment hold the findings of the previous experiments. The baseline and MARS(0.0) models had the same metrics, these models got the highest impact and profit. On the other hand, the standard and MARS(11.4108) models share the lowest impact and profit. It is important to notice that in this experiment the baseline and MARS(0.0) models increased the diversity, this was not evidenced in previous experiments. The maximum obtainable profit under this setup is 6.4593 times the profit obtained by the standard model, while the impact generated to obtain this profit is 0.8536.

3.1.3 AmazonVG2

For the experiments performed on the AmazonVG2 data set, first we used the same predicted matrix R^* as in the experiment of AmazonVG data set. The values in the predicted

matrix are in the range $R^*(u,i) \in [0.5036, 5.736]$. We performed two different experiments with AmazonVG2 data set by modifying the attributes considered in each experiment. In these experiments, we not only considered the relevance of the items and the profit generated, we also considered the price of the items. The variation on the two experiments was the chosen attributes. Consequently, the first experiment considered the relevance of the items, the profit generated by each item, and the inverse of the price $(1/price)$; where items with lower prices had an advantage over items with higher prices. The only difference in the second experiment is that we considered the actual price; where items with higher prices had an advantage over items with lower prices. In each experiment we calculate the metrics for the standard method, the baseline method and various MARS models with different weight distributions. Compared to AmazonVG data set, AmazonVG2 has a variable profit percentage associated to the items in the range of 10 % to 20 %. Then, we can use the price attribute to re-rank the items based not only on the profit, but also on the price of the items. This experiment was designed to test the ability of MARS to consider several attributes. The attribute associated to the relevance of the items for the users increases the accuracy. The attributes associated to the profit generated and the price of each item increase the impact on the user. The values of the weights associated to the profit and the price attributes are similar.

Table 3.7:
Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 4.5$, and taking into account the relevance of the items, the inverse of the price and the profit attributes on AmazonVG2 data set

	Accuracy	Impact	NDCG	RMSE	MAE	Presicion	Profit	Price	Items
Standard	1.0000	0.0000	1.0000	0.0000	0.0000	0.8947	219.98	1263.66	313
MARS(0.96,0.08)	1.0000	0.0000	0.9998	1.0072	0.6569	0.8947	219.98	1263.66	313
MARS(0.89,0.12)	0.9126	0.0874	0.9982	4.5455	2.9347	0.8904	221.00	1265.66	324
MARS(0.89,0.12)*	0.8210	0.1790	0.9944	10.6198	6.5354	0.8734	221.93	1262.40	361
MARS(0.87,0.14)	0.6517	0.3483	0.9820	29.5891	18.3414	0.8416	215.50	1189.65	414
MARS(0.85,0.18)	0.3617	0.6383	0.9446	127.7095	81.9669	0.8113	267.58	1427.30	552
MARS(0.51,0.78)	0.1539	0.8461	0.8044	1405.1599	1164.3168	0.8667	688.37	3777.33	707
MARS(0.0,1.0)	0.1480	0.8520	0.7809	1619.0887	1423.4490	0.7821	325.15	1548.43	647

Table 3.7 presents the results obtained using the inverse of the price attribute. In this experiment, MARS is expected to re-rank the items to obtain a higher profit, but at the same time, the average price of the items should not be high. The company has the benefit of higher profits, while the user has the benefit of lower prices. We used the standard model as benchmark to compare the differences in profit and price with the MARS models, and we also compared between the MARS models. The standard model got an average pro-

fit of $219.98 at a price of $1263.66. MARS(0.51,0.78) had the highest profit and price, $688.37 and $3777.33 respectively. Normally, the highest profit is associated to the highest price, but the setup of this experiment helped control this trade-off. MARS(0.89,0.12) and MARS(0.89,0.12)* models had the same weight distribution, but when we compared the two models, MARS(0.89,0.12)* had a slightly higher profit at a lower price. This results show that MARS is re-ranking based on the conditions given to the method, and it is looking for the highest profit at the lowest price. Accuracy is still affected by the weights given to the relevance of the items. If $\lambda(std)$ decreases, the accuracy decreases; on the other hand, if $\lambda(profit)$ and $\lambda(price)$ increase, the accuracy decreases because relevance is taken less into account. In previous experiments, when the impact increased, the profit also increased. In this experiment there is no direct correlation between impact and profit. For example, MARS(0.51,0.78) had a profit of $688.37 introducing an impact of 0.8461, while MARS(0.0,1.0) had a higher impact of 0.8520 and got a lower profit of $325.15. This is due to the fact that we are not only considering profit anymore, there is the inverse of the price attribute to consider. Generally, higher prices are related to higher profit because the profit is a percentage of the price. When we consider the inverse of the price and the profit, it is like considering opposite attributes.

Table 3.8:
Scores obtained by the standard method, the baseline method and various MARS models, using $T_R = 4.5$, and taking into account the relevance of the items, the price and profit attributes on AmazonVG2 data set

	Accuracy	Impact	NDCG	RMSE	MAE	Presicion	Profit	Price	Items
Standard	1.0000	0.0000	1.0000	0.0000	0.0000	0.8947	219.98	1263.66	313
MARS(0.96,0.08)	1.0000	0.0000	0.9998	1.0325	0.6875	0.8947	219.98	1263.66	313
MARS(0.89,0.12)	0.9129	0.0871	0.9981	4.7432	3.0696	0.9012	263.57	1563.36	324
MARS(0.89,0.12)*	0.8174	0.1826	0.9934	12.9864	7.4606	0.9205	328.46	2001.13	337
MARS(0.87,0.14)	0.6982	0.3018	0.9813	39.7629	22.1155	0.9340	444.14	2722.92	375
MARS(0.84,0.17)	0.4240	0.5760	0.9331	231.4693	134.0688	0.9421	844.11	5093.11	421
MARS(0.0,1.0)	0.1462	0.8538	0.7917	1493.6496	1278.0816	0.9120	1601.67	9337.62	480

Table 3.8 presents the results obtained using directly the price attribute (without inverting it). In this experiment, MARS is expected to re-rank the items to obtain a higher profit, but at the same time, the most expensive items have an advantage over the items with low prices. Then, the profit is going to be high, but the average final price that the users have to pay is also going to be high. We used the standard model as benchmark to compare the differences in profit and price with the MARS models, and we also compared between the MARS models. The standard model got an average profit of $219.98 at a price of $1263.66, while the standard model is the same as the previous experiment. MARS(0.0,1.0) had the

highest profit and price. This model accomplished a profit of $1601.67 at a price of $9337.62. Normally, the highest profit is associated to the highest price, the setup of this experiment made this trade-off more evident. Like in the previous experiment, MARS(0.89,0.12) and MARS(0.89,0.12)* models had the same weight distribution, but when we compared the two models, MARS(0.89,0.12)* had a higher profit at a higher price. This results show that MARS is re-ranking based on the conditions given to the method, and it is looking for the highest profit at the highest price. In this experiment, the accuracy is still affected by the weights given to the relevance of the items. If $\lambda(std)$ decreases, the accuracy decreases; on the other hand, if $\lambda(profit)$ and $\lambda(price)$ increase, the accuracy decreases because relevance is taken less into account. The experiment with the inverse of the price showed no correlation between profit and accuracy. While, in previous experiments with other data sets, when the impact increased, the profit also increased. In this experiment, we again found this correlation between impact and profit. Whenever impact increases, profit also increases. For example, MARS(0.89,0.12) had a profit of $263.57 showing an impact of 0.0871, while MARS(0.0,1.0) had an almost tenfold impact of 0.8538 and got a sixfold profit of $1601.67. In the previous experiment, when we considered the inverse of the price and the profit, it was like considering opposite attributes. But generally, higher prices are related to higher profit because the profit is a percentage of the price. Then, in this experiment, some items have an extra advantage. In specific, items with high prices have an advantage because those items will also have high profit, even if the profit percentage is low.

3.2 RESEARCH QUESTION

What are the parameters or algorithms that could help control the impact on the user introduced by Profit-Aware Recommender Systems when recommending the TopN items to a user?

During this research we found several parameters that we could include on the proposed method to control the impact on the user when recommending the TopN items to a user. The first parameter, which was already defined in [32], is the ranking threshold T_R. This threshold controls the impact by not letting items with low predicted ratings to enter to the final TopN list presented to the user. Another parameter included in the proposed approach was the use of weights to perform the re-ranking process. The weights tell the model how much importance is given to each attribute (i.e. item relevance, profit). The weights control

the impact on the user by re-ranking the items using a weighted rank aggregation algorithm based on the ranking scores of each item attribute. Then, we also found that weighted rank aggregation helps the method to control the impact on the user. Other algorithm used in the proposed method to control the impact is a variant of gradient descent algorithm. We used the variant of gradient descent to find the optimal weights to obtain the desired impact on the user, avoiding having higher impact on the users than desired.

3.3 DISCUSSION

The results of this research have provided a solid foundation on how to control the impact on the user when profit is taken into account for products recommendation. All the experiments have shown that MARS models have the ability to control the impact on the user while increasing the profit by varying the weights associated to the considered attributes. Normally, if a lower impact is desired, we need to give higher weights to the attributes that favor the accuracy. Depending on the parameters given to the model, MARS can achieve several different results in terms of impact and profit. MARS method results may vary between only taking into account the preferences of the users (standard) to only taking into account the most profitable items. The results proved that higher values of $ratio_\lambda$ lead to higher accuracy and lower profit, while lower values of $ratio_\lambda$ produce lower accuracy and higher profit. MARS can control the impact on the user not only by the weights given to the system, but also by adjusting the threshold T_R. The number of items being re-ranked depends on the value of T_R. If $T_R \leq \min R^*$, then all the items are re-ranked. On the other hand, if $T_R > \max R^*$ none of the items are re-ranked. When $\min R^* < T_R \leq \max R^*$, some items are going to be re-ranked. The diversity may be affected by the proposed method, but this depends on the data set being used and the parameters given to the model. In some experiments the diversity increases, and in other experiments the diversity decreases when the model is trained. It is important to notice that the training of the model can be parallelized. In other words, the weights and results can be obtained for each user separately. The model is not needed to be trained with all the users, or the complete matrix. During the experiments we noticed that we could not introduce impact on certain users. This depends on the value of T_R and the predicted ratings of the items for that user ($R^*(u)$). In specific, if $T_R > \max R^*(u)$, then no items are re-ranked and no impact can be introduced to user u.

MARS method depends on the accuracy of the predicted rating matrix R^*. If R^* does not

adequately reflect the user preferences, MARS model will not either. In this work, we assume that the matrix R^* perfectly reflects the user preferences and the accuracy is calculated based on this assumption. Then, in future work, it is important to find a suitable method to predict R^*. Also, a better weights optimization method is needed when multiple attributes are considered. The optimization method proposed in this work divides all the considered attributes into two groups, those that favor the accuracy and those that favor the objective sought by the model (profit in this case). It would be a better approach to optimize each attribute independently. An important part of the MARS method is that the weights can be updated based on the user interaction with the products to obtain more personalized recommendations as the user makes use of the system.

For future work, MARS models can be trained with more and different attributes. Other data sets may be considered in new experiments to obtain insights on how the properties of the data sets affect on the MARS models. Also, other optimization algorithms can be designed to train the model in a better way or to optimize using a multi-objective algorithm. Finally, MARS is not only designed to work with an impact vs. profit trade-off, since other objectives may be defined. For example, we can use the MARS method to increase diversity, to promote certain items, or to give an advantage to a certain item provider by using the corresponding attributes that favor the sought objective.

4 CONCLUSIONS

Nowadays, recommeder systems are being used widely by companies to recommend relevant products to each user and help the user in the decision-making process. These advantages for the user build customer loyalty in a company only if the recommendations are relevant to the user. Some recommeder systems make their recommendations based on the profit generated by each item. This type of recommender systems (PARS) introduce an impact on the user with the recommendations being made. If we are not careful about the impact, customer loyalty can be affected.

This work proposed a recommender system named MARS that re-ranks items based on different attributes selected by the service provider. Specifically, in this work we used the profit of the items in all the experiments, and the price of the items in some experiments. To avoid compromising the relevance of the items being presented, MARS uses a threshold T_R that determines which items are re-ranked. If the item has a low predicted rating (lower than T_R), it is not going to be re-ranked and it is not climbing up in the ranking. MARS uses a score-based weighted rank aggregation method to obtain the final recommendation list. The weights can be personalized for each user, and are updated by an optimization algorithm. MARS is trained by a modified gradient descent algorithm that update the weights to obtain the desired impact on the user. In this way, the model controls the impact on the user so as not to compromise the customer loyalty, while increasing the profit for the company. We got promising results in terms of impact on the user and profit when we implemented MARS on three different data sets under different configurations.

5 REFERENCES

[1] D. Jannach y M. Jugovac, «Measuring the business value of recommender systems,» *ACM Transactions on Management Information Systems (TMIS)*, vol. 10, n.º 4, págs. 1-23, 2019.

[2] D. Jannach y G. Adomavicius, «Recommendations with a purpose,» en *Proceedings of the 10th ACM conference on recommender systems*, 2016, págs. 7-10.

[3] N. Polatidis y C. K. Georgiadis, «Recommender systems: The Importance of personalization in E-business environments,» *International Journal of E-Entrepreneurship and Innovation (IJEEI)*, vol. 4, n.º 4, págs. 32-46, 2013.

[4] B. M. Sarwar, G. Karypis, J. Konstan y J. Riedl, «Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering,» en *Proceedings of the fifth international conference on computer and information technology*, Citeseer, vol. 1, 2002, págs. 291-324.

[5] M. H. Mohamed, M. H. Khafagy y M. H. Ibrahim, «Recommender systems challenges and solutions survey,» en *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, IEEE, 2019, págs. 149-155.

[6] J. Bobadilla, F. Ortega, A. Hernando y A. Gutiérrez, «Recommender systems survey,» *Knowledge-based systems*, vol. 46, págs. 109-132, 2013.

[7] F. Ricci, L. Rokach y B. Shapira, «Introduction to recommender systems handbook,» en *Recommender systems handbook*, Springer, 2011, págs. 1-35.

[8] P. Lops, M. De Gemmis y G. Semeraro, «Content-based recommender systems: State of the art and trends,» *Recommender systems handbook*, págs. 73-105, 2011.

[9] J. B. Schafer, D. Frankowski, J. Herlocker y S. Sen, «Collaborative filtering recommender systems,» en *The adaptive web*, Springer, 2007, págs. 291-324.

[10] R. Burke, «Hybrid recommender systems: Survey and experiments,» *User modeling and user-adapted interaction*, vol. 12, n.º 4, págs. 331-370, 2002.

[11] D. Jannach y G. Adomavicius, «Price and profit awareness in recommender systems,» *arXiv preprint arXiv:1707.08029*, 2017.

[12] G. Adomavicius y A. Tuzhilin, «Context-aware recommender systems,» en *Recommender systems handbook*, Springer, 2011, págs. 217-253.

[13] P. G. Campos, F. Díez e I. Cantador, «Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols,» *User Modeling and User-Adapted Interaction*, vol. 24, n.º 1, págs. 67-119, 2014.

[14] L.-S. Chen, F.-H. Hsu, M.-C. Chen e Y.-C. Hsu, «Developing recommender systems with the consideration of product profitability for sellers,» *Information Sciences*, vol. 178, n.º 4, págs. 1032-1048, 2008.

[15] A. Azaria, A. Hassidim, S. Kraus, A. Eshkol, O. Weintraub e I. Netanely, «Movie recommender system for profit maximization,» en *Proceedings of the 7th ACM conference on Recommender systems*, 2013, págs. 121-128.

[16] A. Das, C. Mathieu y D. Ricketts, «Maximizing profit using recommender systems,» *arXiv preprint arXiv:0908.3633*, 2009.

[17] K. Hosanagar, R. Krishnan y L. Ma, «Recomended for you: The impact of profit incentives on the relevance of online recommendations,» *ICIS 2008 Proceedings*, pág. 31, 2008.

[18] F. Ricci, «Mobile recommender systems,» *Information Technology & Tourism*, vol. 12, n.º 3, págs. 205-231, 2010.

[19] S. Berkovsky, T. Kuflik y F. Ricci, «Mediation of user models for enhanced personalization in recommender systems,» *User Modeling and User-Adapted Interaction*, vol. 18, n.º 3, págs. 245-286, 2008.

[20] P. Pirasteh, D. Hwang y J. J. Jung, «Exploiting matrix factorization to asymmetric user similarities in recommendation systems,» *Knowledge-Based Systems*, vol. 83, págs. 51-57, 2015.

[21] G. Takács, I. Pilászy, B. Németh y D. Tikk, «Investigation of various matrix factorization methods for large recommender systems,» en *2008 IEEE International Conference on Data Mining Workshops*, IEEE, 2008, págs. 553-562.

[22] Y. Koren, R. Bell y C. Volinsky, «Matrix factorization techniques for recommender systems,» *Computer*, vol. 42, n.º 8, págs. 30-37, 2009.

[23] H.-F. Yu, C.-J. Hsieh, S. Si e I. Dhillon, «Scalable coordinate descent approaches to parallel matrix factorization for recommender systems,» en *2012 IEEE 12th international conference on data mining*, IEEE, 2012, págs. 765-774.

[24] Y. Zhou, D. Wilkinson, R. Schreiber y R. Pan, «Large-scale parallel collaborative filtering for the netflix prize,» en *International conference on algorithmic applications in management*, Springer, 2008, págs. 337-348.

[25] Y. Nemati y H. Khademolhosseini, «Devising a Profit-Aware Recommender System using Multi-Objective GA,» *Journal of Advances in Computer Research*, vol. 11, n.º 3, págs. 109-120, 2020.

[26] U. Panniello, S. Hill y M. Gorgoglione, «The impact of profit incentives on the relevance of online recommendations,» *Electronic Commerce Research and Applications*, vol. 20, págs. 87-104, 2016.

[27] U. Panniello, M. Gorgoglione, S. Hill y K. Hosanagar, «Incorporating profit margins into recommender systems: A randomized field experiment of purchasing behavior and consumer trust,» 2014.

[28] L. Akritidis, A. Fevgas, P. Bozanis e Y. Manolopoulos, «An unsupervised distance-based model for weighted rank aggregation with list pruning,» *Expert Systems with Applications*, vol. 202, pág. 117 435, 2022.

[29] E. Dopazo y M. L. Martínez-Céspedes, «Rank aggregation methods dealing with ordinal uncertain preferences,» *Expert Systems with Applications*, vol. 78, págs. 103-109, 2017.

[30] D. J. Dittman, T. M. Khoshgoftaar, R. Wald y A. Napolitano, «Comparison of rank-based vs. score-based aggregation for ensemble gene selection,» en *2013 IEEE 14th International Conference on Information Reuse & Integration (IRI)*, IEEE, 2013, págs. 225-231.

[31] V. Pihur, S. Datta y S. Datta, «Weighted rank aggregation of cluster validation measures: a monte carlo cross-entropy approach,» *Bioinformatics*, vol. 23, n.º 13, págs. 1607-1615, 2007.

[32] G. Adomavicius e Y. Kwon, «Improving aggregate recommendation diversity using ranking-based techniques,» *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, n.º 5, págs. 896-911, 2011.

[33] A. Kouadria, O. Nouali y M. Y. H. Al-Shamri, «A multi-criteria collaborative filtering recommender system using learning-to-rank and rank aggregation,» *Arabian Journal for Science and Engineering*, vol. 45, n.º 4, págs. 2835-2845, 2020.

[34] Y. Shi, M. Larson y A. Hanjalic, «List-wise learning to rank with matrix factorization for collaborative filtering,» en *Proceedings of the fourth ACM conference on Recommender systems*, 2010, págs. 269-272.

[35] B. Lika, K. Kolomvatsos y S. Hadjiefthymiades, «Facing the cold start problem in recommender systems,» *Expert Systems with Applications*, vol. 41, n.º 4, págs. 2065-2073, 2014.

[36] S. Khusro, Z. Ali e I. Ullah, «Recommender systems: issues, challenges, and research opportunities,» en *Information Science and Applications (ICISA) 2016*, Springer, 2016, págs. 1179-1189.

[37] A. R. Hevner, S. T. March, J. Park y S. Ram, «Design science in information systems research,» *MIS quarterly*, págs. 75-105, 2004.

[38] A. Hevner y S. Chatterjee, «Design science research in information systems,» en *Design research in information systems*, Springer, 2010, págs. 9-22.

[39] L. Bunnell, K.-M. Osei-Bryson y V. Y. Yoon, «FinPathlight: Framework for an multiagent recommender system designed to increase consumer financial capability,» *Decision Support Systems*, vol. 134, pág. 113 306, 2020.

[40] P. Hanafizadeh, M. B. Firouzabadi y K. M. Vu, «Insight monetization intermediary platform using recommender systems,» *Electronic Markets*, págs. 1-25, 2021.

[41] R. Miranda, D. Ferreira, A. Abelha y J. Machado, «Intelligent nutrition in healthcare and continuous care,» en *2019 International Conference in Engineering Applications (ICEA)*, IEEE, 2019, págs. 1-6.

[42] R. Baskerville, A. Baiyere, S. Gregor, A. Hevner y M. Rossi, «Design science research contributions: Finding a balance between artifact and theory,» *Journal of the Association for Information Systems*, vol. 19, n.º 5, pág. 3, 2018.

[43] K. Peffers, T. Tuunanen, M. A. Rothenberger y S. Chatterjee, «A Design Science Research Methodology for Information Systems Research,» *Journal of Management Information Systems*, vol. 24, n.º 3, págs. 45-77, 2007. DOI: 10.2753/MIS0742-1222240302. eprint: https://doi.org/10.2753/MIS0742-1222240302. dirección: https://doi.org/10.2753/MIS0742-1222240302.

[44] F. M. Harper y J. A. Konstan, «The movielens datasets: History and context,» *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, n.º 4, págs. 1-19, 2015.

[45] J. Ni, J. Li y J. McAuley, «Justifying recommendations using distantly-labeled reviews and fine-grained aspects,» en *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, págs. 188-197.

[46] Amazon, *How Much Does it Cost to Sell on Amazon?* https://sell.amazon.com/pricing, Accessed: 2022-10-01, 2022.

[47] K. Järvelin y J. Kekäläinen, «Cumulated gain-based evaluation of IR techniques,» *ACM Transactions on Information Systems (TOIS)*, vol. 20, n.º 4, págs. 422-446, 2002.

[48] S. Balakrishnan y S. Chopra, «Collaborative ranking,» en *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, págs. 143-152.

ANNEX

Link to data sets:

https://www.kaggle.com/datasets/juanriofrio/mars-method

Link to Python implementation:

https://github.com/ColdRiver93/MARS/