Quantity or quality? Feature selection techniques to select relevant information in personalized recommender systems

Roger Bagué Masanés
Universidad Internacional Menéndez Pelayo, UIMP - AEPIA
Lleida, España
rogerbm26@gmail.com

***Abstract*—Recommender systems are methods that suggest products or services to users, based on their popularity or the preferences of the users. To do this, they use the information available on platforms in which these items are offered, including the opinions of the users: a simple numerical assessment, which is usually accompanied by a comment and/or a set of pictures. However, these platforms have additional information about the items themselves that sometimes is not used or integrated into recommender systems. This paper focuses on the design and implementation of a personalized recommender system based on the information available about the item in question. The proposed method will be evaluated in a dataset downloaded from TripAdvisor, in which there is information available from restaurants. This information includes, among others: *price range*, types of *cuisines*, *special diets*, *meals*, and other *features* (e.g., free WiFi, outdoor seating, wheelchair accessible). In this context, the use of feature selection techniques is proposed to analyze the impact that each variable has on the personalized recommendations, allowing us to understand the process underlying the recommendation to favour the transparency of the system.**

***Index Terms*—recommender system, feature selection, TripAdvisor, MIM, ReliefF**

I. Introduction

Recommender systems (RS) are software tools and techniques that provide suggestions for items that may be helpful or interesting to users [1, 2, 3]. These suggestions are related to various decision-making processes, e.g., what items to buy, what music to listen to, or what restaurant to eat at.

In recent years, RS have been very important in the research area and industry [4]. There are many fields where RS can be applied since they can be used to maximize the user experience in different domains like music, books, movies, and any type of product that can be sold through the Internet.

RS seek to predict the importance or preference that a user can give to a certain item or group of items [5]. This ability is very important in e-commerce applications, but also in other areas such as expert systems, online dating and social networks. The need for these sites to provide their users with personalized suggestions is not only a desirable feature, but with the large amount of data that they handle and offer, it has become a necessity.

RS have succeeded in changing the way we consume new content and discover new products. One of the clearest examples can be seen on online stores such as Amazon or eBay. With a high level of precision, these websites can provide us with product suggestions adapted to our needs and/or tastes. The same happens with content platforms such as YouTube, Spotify or Netflix. Their recommendations help us discover new videos, artists or series by analyzing our tastes and preferences.

This translates into some general features that the RS can offer us [6]:

- Better satisfaction of customer needs.
- User experience becomes a more pleasant activity since these systems act as a personal assistant that encourages the person to continue discovering elements.
- Provide exceptional efficiency to website conversions.
- Personalized product recommendations bring the customer closer to what they want, improving the chances that they buy or consume the suggested content.
- These systems help us obtain data for statistical reports, and these data can exceptionally feed our machine learning or artificial intelligence models.

There are several types or categories of RS, depending on the information they use. On the one hand, there are personalized RS, which suggest articles or services based on user preferences; and non-personalized RS, which are based on the popularity of the items. On the other hand, collaborative RS use to leverage other users' ratings to provide the recommendations, whilst content-based RS use descriptive keywords associated with the items. More advanced RS use not only users' ratings, but also their reviews and the images they took of the items. Finally, it is worth mentioning that most RS used in practical applications are a combination of these basic models.

In this context, we propose to take into account the users' ratings along with a set of details of the items, which in our case of study are restaurants found on platforms such as TripAdvisor[1]. TripAdvisor is an online platform that recognizes millions of opinions about certain businesses in the tourism industry (e.g., restaurants, hotels, bars, nightclubs, cruises). In this platform, there are a total of 208 different details for

[1]https://www.tripadvisor.com/

each restaurant, within the following categories: *price range*, *cuisine*, *special diets*, *meals*, and *features*.

This large amount of data can be a problem when it comes to algorithm training. To face this problem, we propose to use feature selection (FS) techniques. FS is the process of discovering important features and discarding irrelevant information. The correct selection of features can lead to enhancement of the inductive learner, whether in terms of learning speed, generalization ability, or simplicity of the induced model. In addition, retaining fewer features can lead to reducing measurement costs, and hope for a better understanding of the domain[7].

FS methods are typically presented in three classes [8]:

- **Filter methods** are generally used as a data preprocessing step, as feature selection is performed independently of the machine learning algorithm. The attributes are classified according to the statistical scores that tend to determine the correlation of the attributes to the target variable.
- **Wrapper methods** require a machine learning algorithm and use its performance as an evaluation criterion. These methods search for the features that best suit the algorithm and aim to improve the performance.
- **Embedded methods** combine the qualities of filter and wrapper methods. They are implemented using algorithms that have their built-in feature selection methods.

The Master's Thesis aims to analyze the impact of using the details (e.g., *price range*, types of *cuisines*, etc.) of each restaurant in a personalized RS. In this context, we will demonstrate the importance of performing feature selection to extract the relevant features and gaining insights about which details of the restaurants are the most important for the users. Note that, in huge datasets, the use of FS can result in better results while also gaining efficiency. For this reason, two different FS methods will be compared and their selected features will be used in the RS.

II. Related work

Previous works can be found in the literature with similar motivations and based on the same ideas to a great extent. In this section, we describe some of them.

Afoudi et al. [9] compared different FS techniques used in RS to improve performance. Their proposal is focused on content-based systems and made a restaurant RS that classifies restaurants by priority of order. They used content-based RS because traditional collaborative filtering algorithms did not show satisfactory performance because of data sparsity and contextual data. Moreover, they used other methods to calculate the similarity between items to get rid of the classic cosine method. Finally, they showed that those alternatives improved the performance of their initial system.

Cataltepe et al. [10] combined different types of recommendation methods, such as content-based and collaborative. Their RS takes into account the user behaviour, different types of content features, and other users' habits to predict movie ratings. They used different types of data from movies like the description of the movie, actors that appear there, the director, the year the movie was made, or the genre. With natural language processing techniques they extracted keyword vectors from movie descriptions. Finally, for each user, they used the user's past ratings and also the essence of the characteristics to offer content feature-based user profiles. In addition, they performed FS on these profiles to make them more reliable.

Díez et al. [11] proposed a framework to estimate the authorship for a pair of user and photo. They used restaurant data from six different cities from TripAdvisor. The objective was to provide personalized recommendations to users accompanied by an image that would likely be taken by them, in such a way that the image would act as an explanation of the recommendation.

Blanco-Mallo et al. [12] explored the use of images to model user's tastes, proposing an image-based RS. They referred to the importance of pre-processing and encoding the images, and they used a pre-trained convolutional autoencoder as a feature extractor. Finally, they compared their method with another approach, based on convolutional neural networks, and studied the effect of applying transfer learning.

Ganu et al. [13] pointed out that users are faced with the arduous task of accessing and reading a large number of reviews to discover useful information. Then, the authors identified the topical and emotional information to improve the user's experience of accessing the comments. They focused on improving the accuracy of recommendations in restaurant review scenarios. The method the authors proposed was to predict a rating from text-based reviews and used soft clustering techniques based on the topics and sentiments to group similar users. Their results showed that the use of text information can lead to better predictions compared to the results obtained from the numerical ratings given by users.

The objective of our work is to prove that the use of FS techniques in RS is helpful to understand the process underlying the recommendation to favour the transparency of the system, and may also lead to an increase in performance. Moreover, FS techniques can help to understand which of the restaurant details are most important to users in terms of the rating they give to restaurants. Additionally, we aim to propose a RS that makes a recommendation as good as the one obtained with all the characteristics but using a simpler subset of them.

III. Materials and methods

A. Dataset

The dataset used in this work is a modified version of the one proposed in [11], which contains the reviews from restaurants downloaded from TripAdvisor of the city of Gijón. The adaptation to this dataset includes the addition of the details of the restaurants that it contained. Those details were downloaded with Scrapy[2] framework as part of the present work.

[2]https://scrapy.org/

As mentioned before, the original dataset contains information about restaurants downloaded from TripAdvisor of the city of Gijón [11]. The information available in the dataset that is used in this research is:

- *restauranId*: numeric identifier of the restaurant.
- *userId*: numeric identifier of the user who wrote the review about the restaurant.
- *rating*: discrete numeric variable that represents the score given by the user to the restaurant.

The information added to the original dataset was the result of scraping all the URLs available in the dataset of [11], available in Kaggle[3]. The first information available on the TripAdvisor website for each restaurant is shown in Fig.1, and is composed of the basic details of a given restaurant.

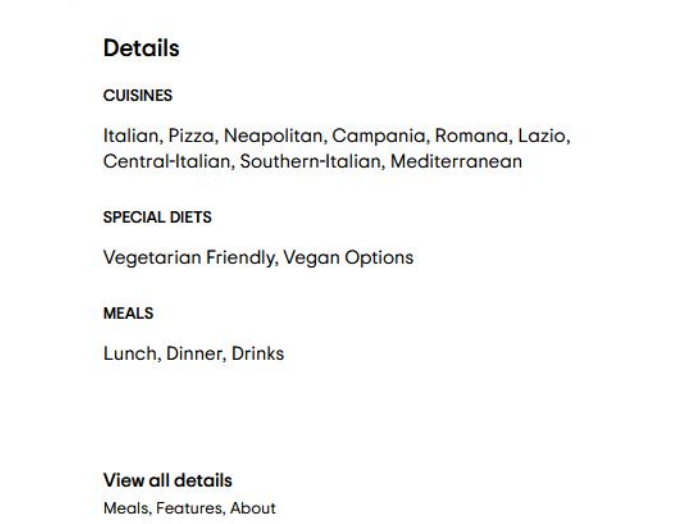

Figure 1. First details available for a sample restaurant.

Fig.2 shows the information that is available when clicking on the option **View all details** of Fig.1. Notice that this information appears in a pop-up window.

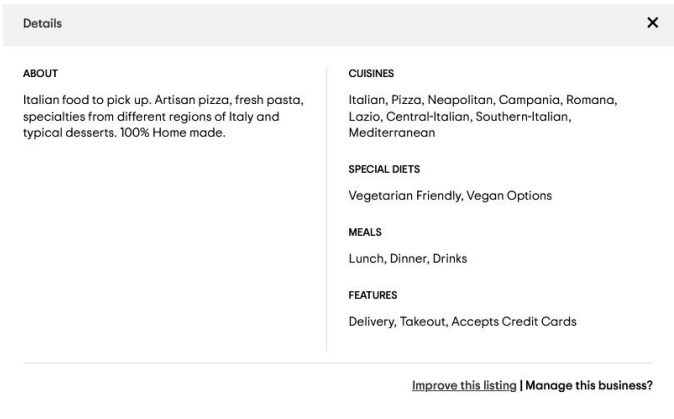

Figure 2. Pop-up with all details of a sample restaurant.

[3]https://www.kaggle.com/chusano/tripadvisor-image-restaurant/version/1

As mentioned above, the tool used to extract all that data is Scrapy, which is an open-source scraping and crawling framework, written in Python. It is a very powerful framework to get information from websites since it allows requests to be made concurrently, which makes it very fast.

Scrapy is used for the *static* part of the HTML in the website, but to manage JavaScript and events (i.e., clicks, scrolls, and others) it was necessary to use Selenium[4] web driver. It is a headless browser that can be configured to run with Firefox, Internet Explorer, Safari, Opera, Chrome, and Edge, and it can send the commands to regions of the website. After the Selenium response (the resulting HTML, for example, a pop-up window), it is passed to Scrapy with a selector.

It is very important to control the times and the waits, there are specific calls and exceptions for timeouts and retries issues, but basically Scrapy, waits for Selenium to manage the event and continues (it is a collaborative task between both). This combination has been mandatory because not all the information was available without interacting with the webpage as can been seen in Fig. 1 and Fig. 2.

1) Analysis of the dataset: In the original dataset, there were a total of 598 different URLs to scrape, but only 529 (88,46%) were available at the time the new data was downloaded. The rest of the URLs were not available, either due to the termination of a business or due to the cancellation of the TripAdvisor subscription. Note that among the 529 restaurants that were available on the platform, 14 do not have any details, representing only 2.65% of the total.

The details of the restaurants are optional and can be grouped into five categories:

- ***Price range*.** It is a range of two prices, the lowest and the highest. Fig. 3 shows a histogram with the top 20 most frequently occurring values in the dataset. Notice that 76.94% of the restaurants do not have this information and, therefore, they are not included in the histogram.
- ***Cuisines*.** Multilabel, maximum selection of five labels out of a total of 156 different options. Fig. 4 shows the top 20 most frequently occurring values in the dataset.
- ***Special diets*.** Multilabel, maximum selection of five labels out of a total of five different options. Fig. 5 shows all values that appear in the dataset.
- ***Meals*.** Multilabel, maximum selection of six labels out of a total of six different options. Figure 6 shows all values that appear in the dataset.
- ***Features*.** Multilabel, maximum selection of 39 labels out of a total of 39 different options. Fig. 7 shows the top 20 most frequently occurring values in the dataset.

For those 529 restaurants available, the dataset also includes their reviews. In particular, there are 48,108 reviews with the following distribution: 40,455 are positive (84.09%) and 7,653 are negative (15.91%). Notice that there is a high level of imbalance in the dataset.

[4]https://www.selenium.dev/

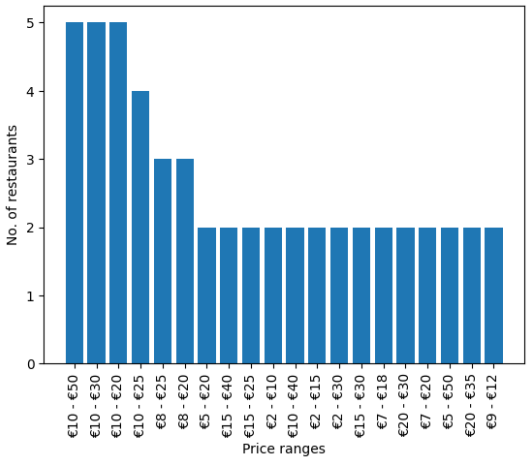

Figure 3. Distribution of Gijón restaurants by *price range*. Note that 76.94% of the restaurants do not include this information.

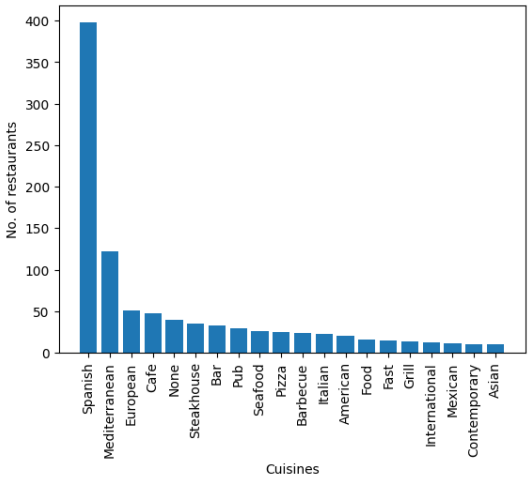

Figure 4. Distribution of Gijón restaurants by types of *cuisines*.

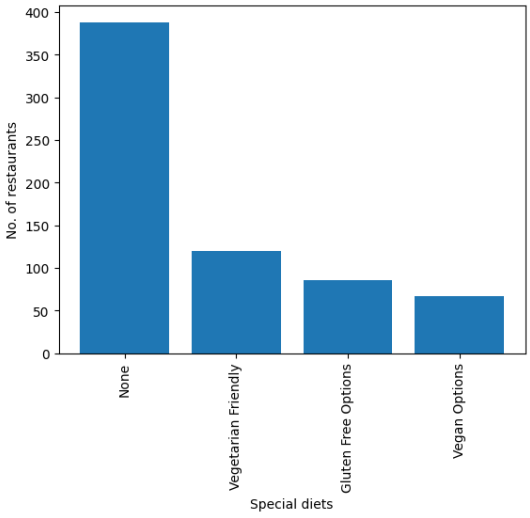

Figure 5. Distribution of Gijón restaurants by *special diets*.

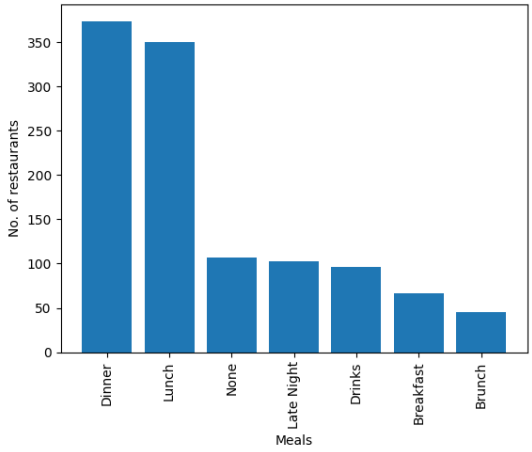

Figure 6. Distribution of Gijón restaurants by types of *meals* offered.

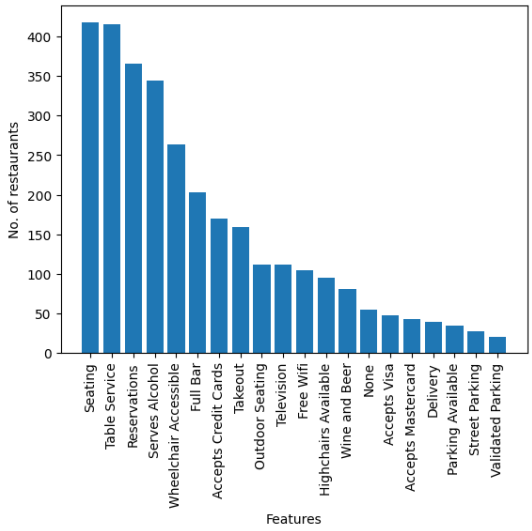

Figure 7. Distribution of Gijón restaurants by other *features*.

2) Data cleaning and preparation: Once the details of all the restaurants were obtained, it is essential to make a small analysis of them to anticipate future problems.

First, the missing data of the details of the restaurants were analyzed. As mentioned before, each restaurant can have information for five categories. The missing data for each category is as follows: the *price range* has 76.36% of missing data, *cuisines* 7.31%, *special diets* 72.80%, *meals* 20.08%, and *features* 10.13%.

As all the different categories are multi-label, except the *price range*, the next step was to determine how to represent them to train the RS. Each multi-label category, with N possible values, must be converted to N binary characteristics. In the case of the *price range*, it is divided into two values, the minimum range and the maximum range, which correspond to the lowest and highest prices, respectively. After this processing, the dataset contains 208 different binary attributes corresponding to all the details that a restaurant can have.

Since there are some categories with many missing values, we decided to represent them as 0 to train the system.

TripAdvisor ratings are represented from 1 to 5 stars but in the original dataset [11] the scores given by the users are integer values between 10 and 50, where 10 represents one star, 20 represents two stars, and so on. This ratio was converted into a new binary attribute *like*, which takes a value of 1 (like) if $rating \geq 30$, and a value of 0 (dislike), if $rating \leq 20$. Columns *restauranId* and userId were encoded as a number starting with 1 and up to N, where N is the number of different restaurants or users.

If there is more than one review from the same user to a certain restaurant, since the publication date of the review is not available in [11], the first review was saved and the rest were discarded.

B. Train/Test split

Once the final dataset was obtained, it was divided into different partitions used for training and evaluation purposes. Given the nature of the dataset, standard divisions with a percentage of samples per partition are not an option in this case since we could have users and/or restaurants in the test set with which the RS has not been trained. For this reason, we created the partitions as described below:

1) Among all the positive reviews of each user, one of them goes to the test set and the rest to the training set. The same procedure is applied to the negative reviews.
2) If there is a user and/or restaurant in the test set that does not appear in the training set, the corresponding review is moved to the training set.

After that, the same procedure is applied to the obtained training set to divide it again into training and validation sets.

Table I shows the distribution of the positive (like) and negative (dislike) classes on the different partitions. As can be seen, the partitions are very unbalanced, which can cause problems in training so that the system cannot predict dislikes because there are very few examples. A solution for this problem is to generate negative sampling, a method which is explained in Section III-C.

Table I
CLASS DISTRIBUTION IN THE THREE PARTITIONS OF THE DATASET.

	Training	**Validation**	**Test**
like	30,643 (82.45%)	3,242 (91.76%)	6,570 (88.65%)
dislike	6,521 (17.55%)	291 (8.24%)	841 (11.35%)
TOTAL	37,164	3,533	7,411

C. Negative sampling

Negative sampling is the process of balancing the training dataset by creating negative reviews, thus having the same or similar number as the positive ones.

The process consists in iterating over all users and checking their number of positive and negative reviews. If the number of negative reviews is greater than the number of positive ones, then it is passed to the next user. Otherwise, if there are more positive reviews than negative, the difference between them is calculated to generate negative reviews. Those negative reviews are randomly generated by choosing restaurants that the user has never visited.

Table II shows the class distribution of the training set before and after applying negative sampling. It can be seen that the modified dataset is balanced. Note that the number of negative reviews is slightly greater than that of positive because there are users who only have negative reviews and, for them, negative sampling was not applied.

Table II
CLASS DISTRIBUTION IN THE TRAINING SET BEFORE AND AFTER APPLYING NEGATIVE SAMPLING (NS).

	Before NS	**After NS**
like	30,643 (82.45%)	30,643 (47.30%)
dislike	6,521 (17.55%)	34,138 (52.70%)
TOTAL	37,164	64,781

D. Feature selection

Datasets can be small while others are tremendously big, especially when they have a large number of features, making them difficult to process. Dealing with high-dimensional datasets can cause:

- The non-relevant or redundant features act as noise so that the model can perform extremely poorly.
- The model takes longer to train.
- Unnecessary resource allocation for these features.

For all these reasons, FS can be a solution when working with high-dimensional datasets.

FS can be defined as the process of selecting the most important and/or relevant features from a dataset, with the aim of improving the learning performance, providing faster and more cost-effective predictors, and producing a better understanding of the process.

Among all the FS methods found in the literature, we used two popular filter methods: mutual information maximization (MIM) and ReliefF, described below. Both methods rank the characteristics according to their importance and also provide a value of this importance. Filter methods were chosen for their independence with the classifier and their low computational cost since the dataset is quite large.

- **Mutual information maximization (MIM)** [14] is based on maximizing the mutual information (MI) between a feature and the class (assuming that the relevant features share information with the class). The MI of two random variables is a quantity that measures the mutual dependence of the two variables; that is, it measures the reduction of the uncertainty (entropy) of a random variable. Note that it is equal to zero only if two random variables are not associate, and higher values mean bigger dependency.
- **ReliefF** [15] is an extension of the original Relief algorithm that can deal with multi-class problems. The main concept behind the Relief algorithm is to estimate the

quality of attributes based on the degree to which the attributes can distinguish instances that are close to each other. ReliefF randomly selects an instance, then searches for the K nearest neighbors of the same class, and then searches for the k nearest misses for each other class. Then ReliefF updates the weight of each attribute, but we average the contribution of all hits and all misses.

E. Recommender system

This section describes the architecture of the proposed RS, which will be used to evaluate the impact of using the restaurant details and FS. The target of the RS is to predict if a user likes or dislikes a certain restaurant. In terms of machine learning, it is a binary classification problem with the user and the restaurant as inputs.

Fig. 8 depicts the proposed architecture for the RS. As can be observed, it receives three input data, *User id*, *Restaurant id* and *Restaurant details*, which are codified as follows:

- **User.** Users are represented by a one-hot codification (embedding) and then mapped into a 64-dimensional vector.
- **Restaurant.**. Restaurants are represented by a one-hot codification (embedding) and then mapped into a 64-dimensional vector.
- **Details.** Details are represented by means of a feature vector.

Once the restaurant and its details are codified, the two vectors are concatenated and a processing block is applied. This **processing block**, which is used repeatedly in the architecture, is composed of a fully connected (FC) layer with L1 regularization [16], a rectified linear unit (ReLU) [17], and batch normalization [18]. The output obtained at this point is concatenated with the user's encoding, followed by another processing block. Next, a sequence of processing blocks is applied up to the final layer that has a FC with the sigmoid activation function, which outputs a probability. Note that the processing blocks halved the number of trainable parameters each time they are applied.

Finally, it is worth mentioning that the binary cross-entropy was used as the loss function and Adam [19] was selected as the optimization algorithm for stochastic gradient descent.

IV. Experimental results

This section includes the results obtained in the two experiments carried out. First, we report the results obtained when applying the FS techniques. Next, we include an analysis focused on the impact of the details in the performance of the RS described in Section III-E.

Note that all the experiments were run in a local environment desktop computer with an Intel Core processor i7-7700 8-cores 3.40GHz, integrated graphics card and 64GB DRAM. The code, which is publicly available on Github[5], was implemented in Python 3.8, with support from the development environment PyCharm 2020.2.2 (Professional Edition), installed on a Windows 10 operating system.

[5]https://github.com/rbague5/TFM

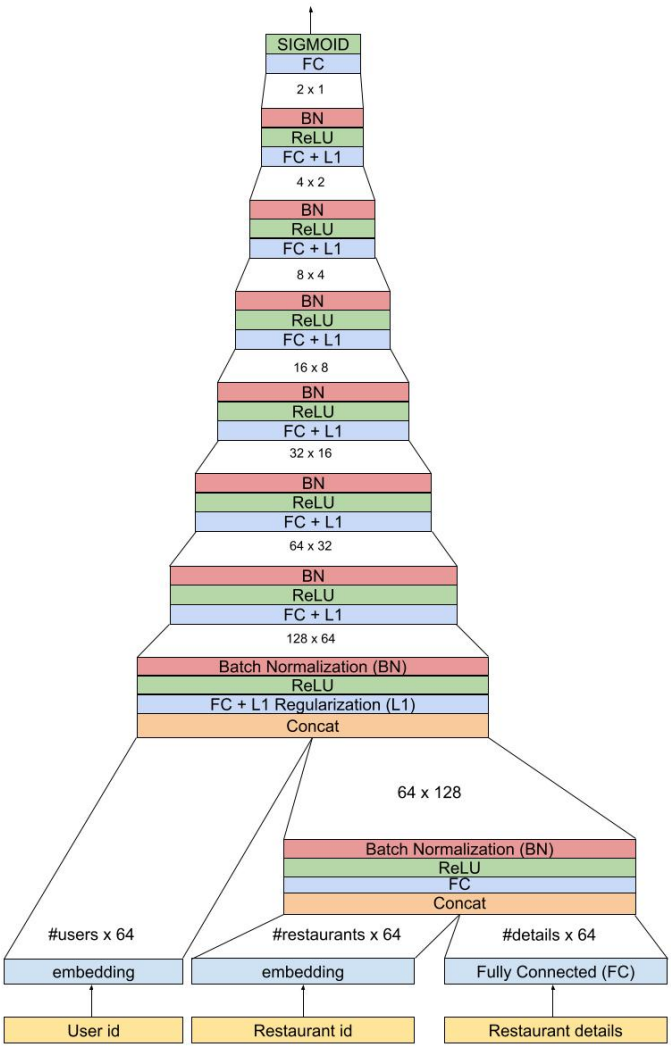

Figure 8. Architecture of the proposed recommender system

A. Feature selection results

Two different FS methods were used to measure the impact of the restaurant details on personalized recommendation, MIM and ReliefF (see Section III-D). Both techniques were applied to the training set described in Section III-B. As a result, the set of details were ordered by a score provided by each FS method.

Tables III and IV show the top 20 details ordered by the scores obtained with MIM and ReliefF, respectively. According to these results, *userId* and *restaurantId* would be the most correlated with the target variable, although they are not relevant to our problem, which is to identify the details that best identify the restaurant. Therefore, for the analysis, both variables were eliminated and the others with a higher value were kept.

As can be observed in Tables III and IV, several attributes are selected by both methods. In this ranking of the 20 most important features, only 12 of them are shared. This may be

Table III
TOP 20 DETAILS SELECTED BY MIM. THE ATTRIBUTES THAT ARE ALSO SELECTED BY RELIEFF ARE REPRESENTED IN BOLD FACE.

Detail	Category	Score
Gluten Free Options	*Special diets*	0.0161
Lunch	*Meals*	0.0160
Dinner	*Meals*	0.0148
Vegetarian Friendly	*Special diets*	0.0106
max_range	*Price range*	0.0104
Highchairs Available	*Features*	0.0102
Vegan Options	*Special diets*	0.0079
min_range	*Price range*	0.0076
Free Wifi	*Features*	0.0049
European	*Cuisines*	0.0048
Private Dining	*Features*	0.0048
Mediterranean	*Cuisines*	0.0039
Reservations	*Features*	0.0036
Accepts Mastercard	*Features*	0.0031
Accepts Visa	*Features*	0.0027
Takeout	*Features*	0.0025
Wheelchair Accessible	*Features*	0.0025
Accepts Credit Cards	*Features*	0.0023
Wine and Beer	*Features*	0.0019
Full Bar	*Features*	0.0019

Table IV
TOP 20 DETAILS SELECTED BY RELIEFF. THE ATTRIBUTES THAT ARE ALSO SELECTED BY MIM ARE REPRESENTED IN BOLD FACE.

Detail	Category	Score
Wheelchair Accessible	*Features*	0.0812
Mediterranean	*Cuisines*	0.0783
Takeout	*Features*	0.0778
Wine and Beer	*Features*	0.0763
Outdoor Seating	*Features*	0.0705
Television	*Features*	0.0690
European	*Cuisines*	0.0682
Drinks	*Meals*	0.0666
Free Wifi	*Features*	0.0663
Late Night	*Meals*	0.0661
Highchairs Available	*Features*	0.0638
Spanish	*Cuisines*	0.0513
Accepts Credit Cards	*Features*	0.0495
Seafood	*Cuisines*	0.0364
Full Bar	*Features*	0.0356
Vegan Options	*Special diets*	0.0335
Accepts American Express	*Features*	0.0279
Digital Payments	*Features*	0.0277
Reservations	*Features*	0.0275
Accepts Visa	*Features*	0.0266

because the two FS methods are based on different metrics: MIM is univariate, while ReliefF is multivariate (it takes into account interactions between attributes).

Focusing on these top 20 selected attributes, we can see that most of them belong to the category called *features*. In the case of MIM it is followed by *special diets*, while in the case of ReliefF the second category more represented is *cuisines*.

- The most important *features* are those related to payment methods and accessibility for people.
- *Special diets* are selected by both FS methods since they are a decisive factor in choosing whether to go to a restaurant or not (e.g., a vegetarian person would not go to a restaurant that only serves meat).
- *Cuisines* are also important in this case. As the data are from a Spanish city, the majority are European, Mediterranean, etc.

Analyzing the execution time, it is worth noting that is a very difference between the two algorithms: MIM only takes 2.90s, while ReliefF about 10 days. The reason is ReliefF is a multivariate method, which takes into account the interactions between characteristics. Moreover, the ReliefF execution time is quadratic to the number of samples while MIM is quadratic to the number of features.

B. Recommender systems results

This experiment aims to analyze the adequacy of the different details on the proposed RS (see Section III-E). For this purpose, we considered not only all the details available in the dataset but also the ones selected by the two FS methods. Additionally, a baseline method was used to measure the relevance of using the details. In this sense, the proposed architecture was slightly modified to be fed with only two input data: *User id* and *Restaurant id*. Consequently, the first processing block after the concatenation between *Restaurant id* and *Restaurant details* was removed, whilst the rest of the blocks are kept.

A grid search was performed to find the best values for the meta-parameters of the proposed RS, using the training and validation sets described in III-B. This search was performed by modifying the embedding size, the batch size, and the learning rate. The meta-parameters that were used for this grid search are:

- **Embedding size:** 64, 128, and 256.
- **Batch size:** 16, 32, and 64.
- **Learning rate:** 0.001, 0.0001, and 0.00001.

As a result, the following configuration was used in all the trained models: an embedding size of 64, a batch size of 16, and a learning rate of 0.0001. The training procedure was monitored with the loss function calculated over the validation set, and an early stopping strategy was used to avoid overfitting.

On the other hand, different performance measures were used for evaluation:

- **True positive rate (TPR)** [20], also known as sensitivity, indicates the ability of our system to correctly predict positive samples.
- **True negative rate (TNR)** [20], also known as specificity, indicates the ability of our system to correctly predict negative cases.
- **The area under the ROC curve (AUC)** [21] is a graphical representation of the TPR (sensitivity) against the FPR ($1-$specificity) for a binary classifier system as the discrimination threshold is varied. AUC measures the area underneath the entire ROC curve.
- **Balanced accuracy (BA)** [22] is an index that can be used when evaluating the quality of a binary classifier, being particularly useful when the classes are unbalanced. It is defined as the average of sensitivity and specificity.

The top 50 details of the ranking obtained with each FS method were used to feed the RS and the results are reported below.

Table V shows the results obtained with MIM, using different numbers of details (selected attributes) from 10 to 50. In general, all combinations produce very similar results with an AUC close to 80%. Using only the top 10 restaurant details achieves the best results in three out of the four metrics considered. In this case, the RS was able to predict 77.99% of positive reviews correctly, while it is only able to predict 50.77% of negative reviews; which implies a balanced accuracy of 64.38%.

Table V
RESULTS OF THE RECOMMENDER SYSTEM WHEN USING THE FEATURES SELECTED BY MUTUAL INFORMATION MAXIMIZATION (MIM). BEST RESULTS ARE IN BOLD.

	10 details	**20 details**	**30 details**	**40 details**	**50 details**
TPR	**0.7799**	0.7190	0.7149	0.7502	0.7326
TNR	0.5077	0.5303	**0.5363**	0.4851	0.5113
AUC	**0.7972**	0.7708	0.7805	0.7724	0.7703
BA	**0.6438**	0.6247	0.6256	0.6177	0.6219

Table VI shows the results obtained with ReliefF, using different numbers of details (selected attributes) from 10 to 50. As happened in Table V, all results are close to 80%. The best results in terms of TNR and BA are obtained with only 10 details, while the best results in terms of TPR and AUC are obtained with 20 details. The one using 10 details is considered the most appropriate because it is more balanced and it better predicts the negative cases, as well as having the benefit of using fewer features.

In this case, the system can correctly predict 69.09% of positive reviews and 55.77% of negative reviews, obtaining a balanced accuracy of 62.43%

Table VI
RESULTS OF THE RECOMMENDER SYSTEM WHEN USING THE FEATURES SELECTED BY RELIEFF. BEST RESULTS ARE IN BOLD.

	10 details	**20 details**	**30 details**	**40 details**	**50 details**
TPR	0.6909	**0.7650**	0.7336	0.7282	0.7014
TNR	**0.5577**	0.4423	0.5030	0.5089	0.5434
AUC	0.7728	**0.7905**	0.7796	0.7759	0.7700
BA	**0.6243**	0.6037	0.6183	0.6185	0.6224

Once the performance of the proposed RS was analyzed with the different subsets of features returned by each FS method, we compared them with the baseline and with an approach that does not use FS (i.e., it uses all the available details). Table VII shows this comparison among the four options, after selecting the most competitive number of details for each FS method.

The first thing that can be appreciated is that using all details improves the baseline in only two of the four metrics (TNR and BA), the baseline is the best option for predicting positive reviews while its performance decays in predicting the negative ones. Alternatively, it can be seen that the use of details is relevant because the best results are achieved using them, and improves when performing a process of feature selection.

Table VII
SUMMARY OF THE RECOMMENDER SYSTEM RESULTS. BEST RESULTS ARE IN BOLD.

	Baseline	**All details**	**MIM (10 *)**	**ReliefF (10 *)**
TPR	0.7470	0.7298	**0.7799**	0.6909
TNR	0.4756	0.5018	0.5077	**0.5577**
AUC	0.7919	0.7749	**0.7972**	0.7728
BA	0.6113	0.6158	**0.6438**	0.6243

* Number of details selected by the FS method in the best case

The RS trained with the best configuration of ReliefF achieves the best results in terms of TNR, which is remarkable given that it is the minority class and often difficult to predict. More specifically, can correctly predict 55.77%, but it is the worst predicting positive reviews, with 69.09% of correct predictions.

On the other hand, the best results in the other three metrics are achieved by the 10 details selected by MIM, which can be highlighted as the most competitive approach for the problem at hand. It is the best approach for predicting positive reviews, with a success of 77.99%. This result and a TNR of 50.77% lead to an AUC of 79.72%.

Note that MIM obtains these results using only 10 of 208 available details, thus generating a much simpler and easier interpretation of the model. These 10 most important details, as can be seen in Table III are: Gluten Free Options, Lunch, Dinner, Vegetarian Friendly, max range, Highchairs Available, Vegan Options, min range, Free Wifi, and European. These attributes are related with *special diets*, the restaurant's *price range*, the types of *cuisines* and whether they have Highchairs Available.

V. CONCLUSIONS

Recommender systems often have to deal with large datasets, both in the number of features and samples. For this reason, we propose the use of feature selection techniques to obtain simpler and more explainable models, without losing accuracy in the recommendation.

This research work is focused on evaluating the use of feature selection in the context of personalized recommendations. We started by analyzing whether the use of the details of the restaurants can be useful. Therefore, we proposed a recommender system that works as follows: for each (user, restaurant) pair, the RS predicts whether the user likes this restaurant or not. Given that the number of details is very large, we used two FS methods to analyze their relevance. The experimentation carried out demonstrated the competitiveness of the prediction results, showing that feature selection can improve the prediction of positive cases by 5.01% (compared to the case of using all the details) and negative cases by 0.59%, using only 4.80% of all available details.

As future work, there are two main lines of development. On the one hand, to evaluate the proposed methods on larger datasets from other cities, in such a way that the RS will have more data to be trained and more diversity of users and tastes. On the other hand, try to also use other data that the users have, such as the text reviews they wrote or the images that they took in the restaurants. Combining all this information could lead to a system with more characteristics in which feature selection techniques could have even more relevance than in the present work.

REFERENCES

[1] T. Mahmood and F. Ricci, "Improving recommender systems with adaptive conversational strategies," in 20th ACM Conference on Hypertext and Hypermedia, 2009, pp. 73–82.

[2] P. Resnick and H. R. Varian, "Recommender systems," Communications of the ACM, vol. 40, no. 3, pp. 56–58, 1997.

[3] R. Burke, "Hybrid web recommender systems," The Adaptive Web, pp. 377–408, 2007.

[4] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," Knowledge-based Systems, vol. 46, pp. 109–132, 2013.

[5] F. O. Isinkaye, Y. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," Egyptian Informatics Journal, vol. 16, no. 3, pp. 261–273, 2015.

[6] GraphEverywhere, "Sistemas de recomendación: Qué son, tipos y ejemplos," 2019. [Online]. Available: https://www.grapheverywhere.com/sistemas-de-recomendacion-que-son-tipos-y-ejemplos/

[7] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," Knowledge and Information Systems, vol. 34, no. 3, pp. 483–519, 2013.

[8] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," Computers & Electrical Engineering, vol. 40, no. 1, pp. 16–28, 2014.

[9] Y. Afoudi, M. Lazaar, and M. Al Achhab, "Impact of Feature selection on content-based recommendation system," in International Conference on Wireless Technologies, Embedded and Intelligent Systems. IEEE, 2019, pp. 1–6.

[10] Z. Cataltepe, M. ULUYAĞMUR, and E. TAYFUR, "Feature selection for movie recommendation," Turkish Journal of Electrical Engineering & Computer Sciences, vol. 24, no. 3, pp. 833–848, 2016.

[11] J. Díez, P. Pérez-Núnez, O. Luaces, B. Remeseiro, and A. Bahamonde, "Towards explainable personalized recommendations by learning from users' photos," Information Sciences, vol. 520, pp. 416–430, 2020.

[12] E. Blanco-Mallo, B. Remeseiro, V. Bolón-Canedo, and A. Alonso-Betanzos, "On the effectiveness of convolutional autoencoders on image-based personalized recommender systems," in Multidisciplinary Digital Publishing Institute Proceedings, vol. 54, no. 1, 2020, p. 11.

[13] G. Ganu, Y. Kakodkar, and A. Marian, "Improving the quality of predictions using textual information in online user reviews," Information Systems, vol. 38, no. 1, pp. 1–15, 2013.

[14] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," Physical review E, vol. 69, no. 6, p. 066138, 2004.

[15] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of ReliefF and RReliefF," Machine Learning, vol. 53, no. 1, pp. 23–69, 2003.

[16] M. Y. Park and T. Hastie, "L1-regularization path algorithm for generalized linear models," Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 69, no. 4, pp. 659–677, 2007.

[17] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in Icml, 2010.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in International Conference on Machine Learning. PMLR, 2015, pp. 448–456.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[20] P. A. Flach, "ROC analysis," in Encyclopedia of Machine Learning and Data Mining. Springer, 2016, pp. 1–8.

[21] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," Pattern Recognition, vol. 30, no. 7, pp. 1145–1159, 1997.

[22] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," in 20th International Conference on Pattern Recognition. IEEE, 2010, pp. 3121–3124.