1. Building a rating matrix.

In this research, the focus is building a Collaborative filtering (CF) recommender systems which uses given rating data for many items by a lot of users for creating a top-N recommendation list for active users.

Generally, there is a set of users U = {u1, u2, . . . , um} and a set of items I = {i1, i2, . . . , in}. A m × n matrix is used to store ratings R=(rjk), where each row represents a user (uj), $1 \leq k \leq n$ and each column represent items ik ($1 \leq k \leq n$). Each row contains as well the ratings of user uj(rjk) for item ik.[1]

Many of the algorithms use ratings on a scale, for example 1 star to 5 starts. In these cases the Recommender systems rely on a high quality explicit feedback from the users which show their interest into certain products. For example, Yelp collects star ratings for the businesses listed, where users express their preferences for the quality of the service they received. Yet, in some instances the explicit feedback cannot be obtained. Therefore, implicit feedback must be used, and it shows the preference by observing user history and behavior. An example of implicit feedback is the purchase history.[2]

The dataset provided by Santander bank is another case of implicit data where each item can have a value of 0 or 1, where 1 means that the user bought the product(has a preference for it) and 0 means that the user doesn't prefer the product.

---

[1] Hahsler, M. (2015*). recommenderlab: A Framework for Developing and Testing Recommendation Algorithms*. Retrieved from https://rdrr.io/cran/recommenderlab/f/inst/doc/recommenderlab.pdf

[2] Hu, Y. , Koren, Y. and Volinsky, Y. (2008).*Collaborative Filtering for Implicit Feedback Datasets*. Eighth IEEE International Conference on Data Mining, Pisa, 2008, pp. 263-272.
doi: 10.1109/ICDM.2008.22

To obtain a rating matrix in the case of Santander bank, the data.table package is used to extract the Id column and the product columns, resulting a data frame with the Id's plus the 1 and 0 ratings for the products, which it will be transformed in a matrix and then coerced in **binaryRatingMatrix** format required by the **recommenderlab** package, used later on in building the recommender systems.

2. Creating the Collaborative Filtering Recommender Systems

In this paper, two collaborative filtering recommender systems will be created by using the package **recommenderlab** :Item-based Recommender System (IBCF) and a Popular Recommender System and they will be compared with a Random Recommender System.

2.1 Item-based collaborative filtering

Item-based collaborative filtering uses the relationship between items deduced from the rating matrix to make recommendations. The main idea behind this approach is that users will choose products that are similar to other products they like. The first step is to calculate a similarity matrix that contains the similarity between all pairs of items. The second step is to make recommendations, where the most similar items to the items that were already rated by user will be recommended.

For the Santander dataset it will the Recommender(data, method, parameters) function from recommenderlab package will be used, which creates a recommender object. The function will use the training data to train the algorithm, based on a "topNlist" method. The function uses

the “cosine” similarity to identify 50 neighboring items (value that can be changed) to base their recommendation on.

The recommender object will be called in the predict() function (part of recommenderlab package) which uses the unkown(test data) to make recommendation. The following parameters will be used for the predict function: “topNlist” method and a n= 5 items that the IBCF recommender systems has to return as recommendations.

2.2 Popular Recommender System

The Popular Recommender System uses the same Recommender function to build the recommender object and the predict function, which returns as recommendation the most popular products (5 products in this case).

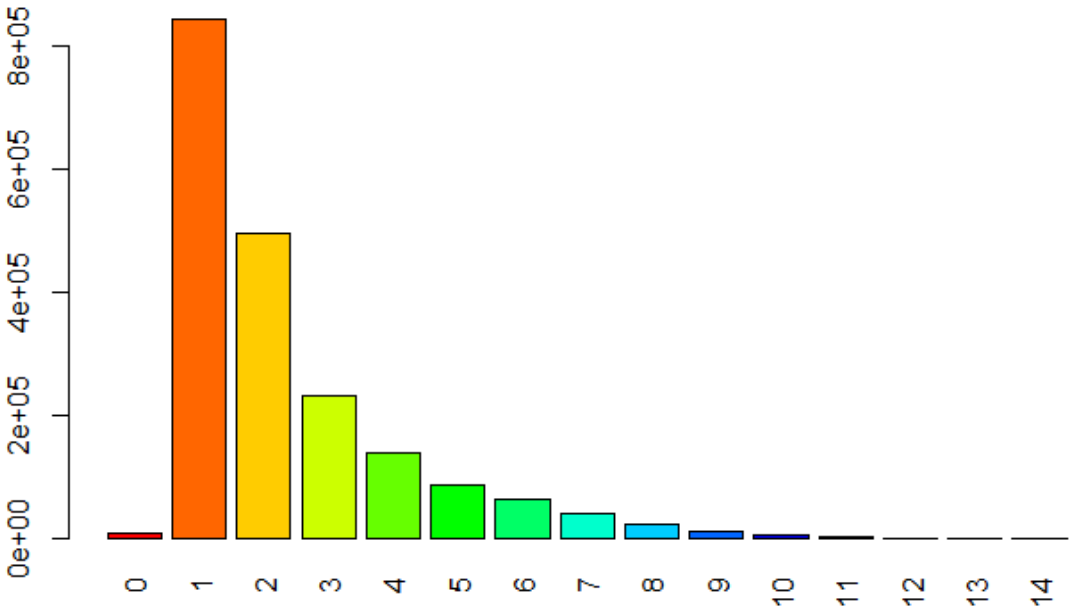

Figure 7. The representation of the number of products per customer

The third recommender system makes random recommendations. As with the previous algorithms, the “RANDOM” method is chosen with the Recommender function and the number of recommendations is placed at 5.

3 Evaluating and testing the models

For model evaluation, in the recommenderlab package, an evaluation scheme can be created and called in the Recommender object. It uses the training data to evaluate the respective model along with one of the following methods: k-fold cross validation , bootstrap sampling, simple split.

For the models built in this study, the evaluation scheme is created using split method with 1 runs which divides the Santander dataset in training set (70%) and is referred as known data in the algorithm and 30 % test set.

Testing inside the evaluation scheme is performed through All-but-1 protocol. With this protocol the algorithm sees all-but-1 rating withheld for the test user.

To obtain the evaluation metrics, the calcPredictionAccuracy() function is used. It has among it’s arguments the evaluation scheme built before, the list of recommendations as a result of applying the predict function and the unknow data saved from training test for evaluating the models.

The evaluation metrics obtained are: TP,TN,FP,FN, TPR, FPR, Precision and Recall. They will be used to calculate mean absolute error and accuracy for each model and all the information will be saved in a data frame.

To determine the best performing algorithm and to better understand how the results will be affected if the number of recommended items will change, another evaluation scheme is created using the cross-validation method. The evaluate() function included in the recommenderlab will run the list of provided algorithms and obtain as a result the evaluation metrics.

The next step is to plot the results acquired before by using the evaluate() function and the output is the representation of the ROC curves for the provided recommender systems. In this manner the algorithms can be easily compared, as the bigger the area under the ROC curve, the better the performance of the algorithm.