A recommendation system for car insurance

Laurent Lesage[1,2] ⬤ **· Madalina Deaconu**[2] **· Antoine Lejay**[2] **· Jorge Augusto Meira**[1] **· Geoffrey Nichil**[3] **· Radu State**[1]

Abstract

We construct a recommendation system for car insurance, to allow agents to optimize up-selling performances, by selecting customers who are most likely to subscribe an additional cover. The originality of our recommendation system is to be suited for the insurance context. While traditional recommendation systems, designed for online platforms (e.g. e-commerce, videos), are constructed on huge datasets and aim to suggest the next best offer, insurance products have specific properties which imply that we must adopt a different approach. Our recommendation system combines the XGBoost algorithm and the Apriori algorithm to choose which customer should be recommended and which cover to recommend, respectively. It has been tested in a pilot phase of around 150 recommendations, which shows that the approach outperforms standard results for similar up-selling campaigns.

Keywords Recommendation system · Up-selling · Car insurance · XGBoost algorithm · Apriori algorithm

1 Summary

Global purpose In this paper, we propose a recommendation system built for a better customers' experience, by suggesting them the most appropriate cover in time. The requirement for this system is to perform a more efficient up-selling than classic marketing campaigns. Recently, the applicability of machine learning algorithms have become very popular in many different areas of knowledge leading to learn up-to-date advanced patterns from customers' behaviour and consequently target

✉ Laurent Lesage
laurent.lesage@uni.lu

1 University of Luxembourg, 29, Avenue JF Kennedy JFK Building, E02-206, 1855 Luxembourg, Luxembourg

2 Université de Lorraine CNRS, Inria, IECL, 54000 Nancy, France

3 Foyer Assurances, 12, Rue Léon Laval, 3372 Leudelange, Luxembourg

customers more accurately. In the context of recommendation systems, such algorithms generate automatically commercial opportunities suited to each customer.

Purpose: up-selling Our recommendation system is currently in use by Foyer Assurances[1] agents. Our goal is to support the agents that are and will continue to be the best advisers for customers, due to their experience and their knowledge of their portfolio. In short, our tool helps them by automatically selecting from their large portfolios the customers most likely to augment their insurance coverage, in order to optimize up-selling campaigns for instance. Thus, an insurance company using this solution could combine advantages from both data analysis and human expertise. Agents validate if the recommendations from our system are appropriate to customers and make trustworthy commercial opportunities for them. The recommendation system is also planned to be integrated in customers' web-pages, in order to provide them a personalized assistance online.

Main applications of recommendation systems Recommendation systems are currently adopted in many web applications. They offer a huge amount of products with daily use (e.g. e-commerce websites, music and video streaming platforms), in order to make customer's decision-making easier and tackle problems related to over-choice. For most famous platforms, such as Amazon and Netflix, users must choose between hundreds or even thousands of products and tend to lose interest very quickly if they cannot make a decision (see [6]). Recommendation systems are then essential to give customers the best experience.

First type of recommendation system Collaborative filtering is the first category of recommendation systems (see [2]). It consists in formulating recommendations by filtering information from many viewpoints or data sources. The first subset of collaborative filtering techniques is the so-called memory-based approach. This type of model compiles similarities and distances between users or items, from ratings given by users to items, or lists of items purchased by each user if there are no ratings. The idea is to identify for a user A either the most similar user B, then recommend to user A items that were already purchased by user B (User-Based Collaborative Filtering: UBCF, see [16]), or items that are the most similar to items user A has already subscribed to (Item-Based Collaborative Filtering: IBCF, see [13]). The second subset of collaborative filtering techniques is the so-called model-based approach, with data mining or machine learning algorithms. A classic model is based on matrix factorization, whose objective is to decompose the user-item interaction matrix (which contains ratings given by users to items), into the product of two matrices of lower dimensions. The main matrix factorization algorithm is Singular Value Decomposition (SVD) (see [4]).

Second type of recommendation system The second category of recommendation systems is the so-called content-based filtering. They analyze information about description of items and compile recommendations from this analysis. The main data source is text documents detailing content of items. A classic approach is Term Frequency–Inverse Document Frequency (TF–IDF, see [11]). Term Frequency

---

[1] Foyer Assurances is leader of individual and professional insurance in Luxembourg.

counts the number of times a term occurs, while Inverse Document Frequency measures how rare a term is and how much information a term provides.

Third type of recommendation system The third category is the so-called hybrid filtering. It consists in mixing the two previous approaches and requires a huge amount of complex data. Deep learning techniques, which could be used for every type of recommendation system, are the most frequent approach to perform hybrid filtering. Given the tremendous improvement of computers' performances in the past few years, deep learning techniques deal with massive information and unstructured data. The survey in [15] lists the different deep learning techniques applied to recommendation systems, useful when dealing with sequences (e.g. language, audio, video) and non-linear dependencies.

Specific context of insurance However, most of the algorithms we have described previously, which are appropriate for large-scale problems and to suggest the next best offer, would not fit for insurance covers recommendation. Indeed, the insurance context differs by three major particularities (see also [12]).

Property 1 Data dimensions: *the number of covers is limited to a small number* (*e.g.* 10–20) *of guarantees. In comparison with thousands of books or movies proposed by online platforms*, *dimensions of the problem are reduced.*

Property 2 Trustworthiness: *insurance products are purchased differently from movies*, *books and other daily or weekly products. Frequency of contacts between an insurance company and customers is reduced since policyholders modify their cover rarely. Therefore*, *a high level of confidence in recommendations for insurance customers is needed. While recommending a wrong movie is not a big deal since the viewer will always find another option from thousands of videos*, *recommending an inappropriate insurance cover could damage significantly the trust of customers in their insurance company.*

Property 3 Constraints: *while any movie or any book could be enjoyed by anyone* (*except for age limit*), *several complex constraints exist when a customer chooses his cover. For instance*, *some guarantees could have an overlap*, *or some criterion linked to customers' profile* (*i.e. age limits*, *no-claims bonus level*, *vehicle characteristics*, *etc.*).

Work related to insurance recommendation systems There are few papers about insurance recommendation systems. In [10], the authors present a system built for agents to recommend any type of insurance (life, umbrella, auto, etc.) based on a Bayesian network. The results of the pilot phase based on the recommendation system from [10] will be used as a baseline for our system. Moreover, properties and particularities of a recommendation system for insurance are listed in [12], which shows the efficiency of a recommendation system for call-centers. A survey of recommendation systems applied to finance and insurance is proposed in [17]. This study lists two other engines for health insurance, whose specificity differs from car insurance.

Contributions The major contributions of this paper are to:

1. propose an architecture suitable to an insurance context and different from classic approaches, by associating a probability of accepting a recommendation to the next best offer,
2. back-test the recommendation system with relevant indicators including a comparison with classical models,
3. present a recommendation system whose results, on a pilot phase including around 150 recommendations, are above standard rates of acceptance.

Main result The recommendation system gets an acceptance rate of 38% during its pilot phase, which is a promising result since classic rates for such a campaign are around 15% (see [10] and Remark 2). In Sect. 4, we suggest enhancements which could allow us to improve this rate.

Plan The remainder of this paper is organized as follows. We propose a suitable approach in Sect. 2, in accordance with the three properties described previously. Then we present the results of back-testing and a pilot phase in Sect. 3 and conclude on future work on the recommendation system in Sect. 4.

2 Recommendation system: architecture and assumptions

The following section presents the global architecture of the recommendation system and the main underlying assumptions.

2.1 Context and objective

The recommendation system that we propose focuses on customers who subscribed to a car insurance product proposed by Foyer. Each and every vehicle from the Luxembourgian car fleet which is driven on public roads (over 400 k vehicles all around the country) has to be insured and covered by the third party responsibility guarantee at least. Foyer is the leader of car insurance in Luxembourg, having a market share of 44% in non-life insurance. Foyer's car insurance product is characterized by a set of guarantees, with a particular structure. In this sense, the customers must select standard guarantees (including third party liability), and could add optional guarantees. The structure of the product, i.e. the proposed guarantees, has not changed for ten years. Given these guarantees, the customers could be covered for theft, fire, material damage, acts of nature, personal belongings, etc. Each customer has an existing cover, which corresponds to a subset of guarantees. The objective of the recommendation system is to assign to each customer the most relevant additional guarantee.

In this paper, we will assume that time is defined as a fraction of years. Let us introduce some notations.

Notation 1 We consider:

1. N the number of customers who subscribed to the car insurance product.
2. M the number of guarantees available for the car insurance product. M_s is the total of standard guarantees and M_o the total of optional guarantees. This leads to:

$$M = M_s + M_o. \tag{1}$$

3. $\mathcal{U}$ the set of the N customers who subscribed to the car insurance product:

$$\mathcal{U} = \{u_1, u_2, \ldots, u_N\}, \tag{2}$$

   where u_i is the customer number i.
4. $\mathcal{G}$ the set of M available guarantees:

$$\mathcal{G} = \{g_1, \ldots, g_M\}, \tag{3}$$

   where $g_1, \ldots, g_M$ are the guarantees. $\mathcal{G}$ is split into two disjoint subsets, $\mathcal{G}_s$ the subset of standard guarantees and $\mathcal{G}_o$ the subset of optional guarantees:

$$\mathcal{G} = \mathcal{G}_s \cup \mathcal{G}_o, \tag{4}$$

$$\mathcal{G}_s = \{g_1, \ldots, g_{M_s}\}, \tag{5}$$

$$\mathcal{G}_o = \{g_{M_s+1}, \ldots, g_{M_s+M_o}\}. \tag{6}$$

5. f_t the function which assigns each customer u_i, $i \in [\![1;N]\!]$, to his existing cover $f_t(u_i)$ at time t:

$$f_t : \mathcal{U} \longrightarrow \mathcal{P}(\mathcal{G}), \tag{7}$$

   where $\mathcal{U}$ and $\mathcal{G}$ are defined by Eqs. (2) and (3) respectively, and $\mathcal{P}(\mathcal{G})$ is the set containing every subset of $\mathcal{G}$. Since each customer must select at least one guarantee from $\mathcal{G}_s$, $\forall i \in [\![1;N]\!], f_t(u_i) \cap \mathcal{G}_s \neq \emptyset$. We also consider $f_t(u_i)^c$, the subset of guarantees u_i did not subscribe to, i.e.: $f_t(u_i) \cup f_t(u_i)^c = \mathcal{G}$.
6. Φ the function which assigns each customer u_i to the index $\Phi(u_i)$ of the guarantee $g_{\Phi(u_i)}$ suggested by the recommendation system:

$$\Phi : \mathcal{U} \longrightarrow [\![1;M]\!], \tag{8}$$

   where $\mathcal{U}$ is defined by Eq. (2). Then $g_{\Phi(u_i)} \in f_t(u_i)^c$, i.e. the guarantee recommended does not belong to the current customer's cover.

Given Property 2, the probability to accept the recommendation should be estimated as well, in order to have a confidence level before sending the recommendation.

Before formulating the objective, we have to fix a parameter: the temporal horizon to accept a recommendation. Beyond this limit, the recommendation is considered as rejected.

Assumption 1 The temporal horizon to accept a recommendation is fixed to 1 year.

This length is chosen to allow customers to have time to weigh up the pros and cons of the recommendation, in accordance with Property 2.

Notation 2 We consider:

$$\boldsymbol{p}_t = (p_t^i)_{i\in[\![1;N]\!]} = \left(\mathbb{P}\Big[g_{\Phi(u_i)} \in f_{t+1}(u_i) | g_{\Phi(u_i)\notin f_t(u_i)} \Big] \right)_{i\in[\![1;N]\!]}, \tag{9}$$

the vector containing the probabilities for each customer to accept the recommendation between times t and $t + 1$, where t is in years and where $f_t(u_i)$ is defined by Eq. (7).

Finally, the recommendation system aims to estimate Φ and $\boldsymbol{p}_t$, in order to evaluate every couple:

$$\{g_{\Phi(u_i)}, p_t^i\}, i \in [\![1;N]\!], \tag{10}$$

where $g_{\Phi(u_i)}$ and p_t^i are defined by Eqs. (8) and (9) respectively.

2.2 The model

We propose the following approach to build the targeted recommendation system, illustrated by Fig. 1. After aggregating the different data sources (step A), we perform feature engineering (step B). Assumption 2 is then made, illustrated by the separation of steps C1 and C2.

Assumption 2 For each customer u_i, we consider that p_t^i and $g_{\Phi(u_i)}$, defined by Eqs. (9) and (8) respectively, are independent.

This strong assumption is motivated by the fact that the objective is to target the right amount of customers most likely to add a guarantee and to avoid non converted opportunities, instead of simply offering the next best offer for everyone (Property 2). Indeed, since we focus on a small amount of covers (Property 1), the Apriori algorithm (see Sect. 2.2.4) and other similar algorithms tested are sufficient to predict which cover/guarantee should be added to current covering. Then, supervised learning on past added covers allows us to be accurate on which customers we should address the recommendations, by integrating features describing their profiles and their current cover. Algorithms used in C1 and C2 are chosen by comparing different methods and picking the best according to a specific back-testing (see Sects. 3.2 and 3.3, respectively). To validate the Assumption 2 a posteriori, we separated the customers into two sub-populations: customers with the lowest and those with the highest probabilities of adding a guarantee, calculated in step C1. Then we compared the accuracy of the Apriori algorithm for theses two sub-populations. This study is presented in Sect. 3.3.

After steps C1 and C2, business rules are applied to tackle Property 3. Then the recommendation system outputs the final list of recommendations, ordered by
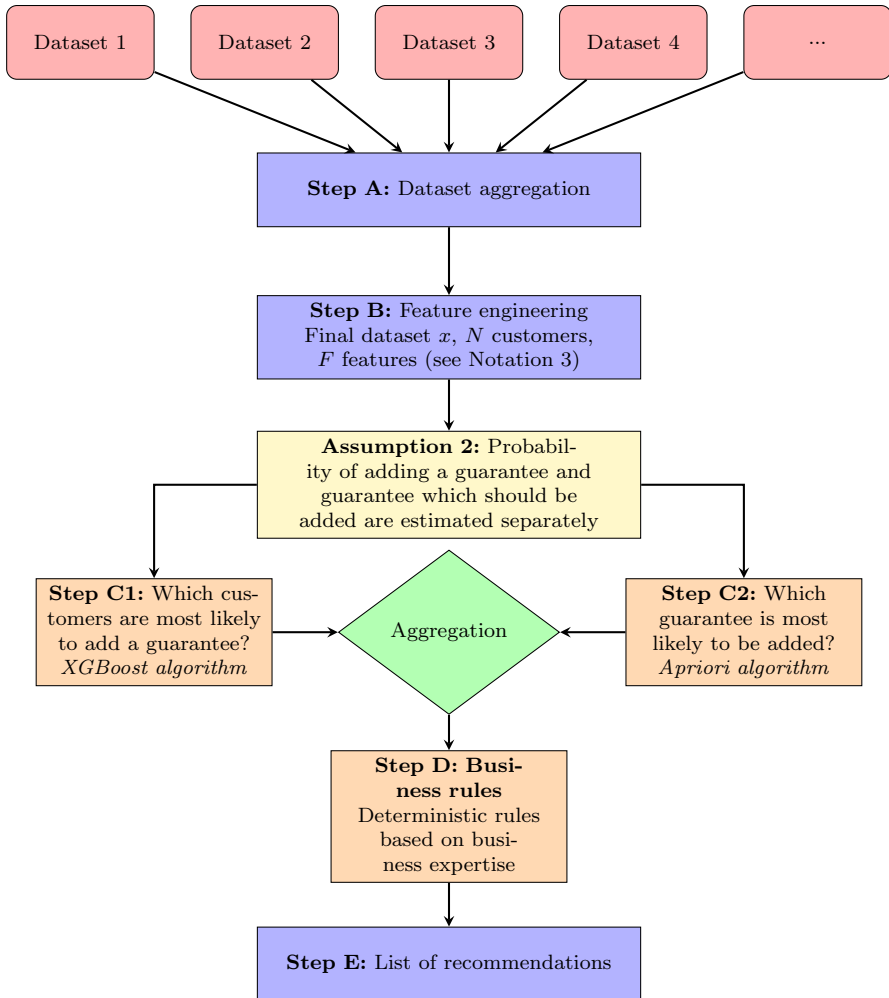
Fig. 1 Global architecture of the recommendation system

decreasing probability of adding a cover. The following subsections describe each step separately.

2.2.1 Step A: dataset aggregation

We built an unique dataset from multiple internal data sources. This dataset includes information about current customers' car policies (current cover, vehicle's characteristics, premium amounts), other insurance products subscribed (home, health, pension, savings), customers' characteristics, information about contacts between customers and the insurance company (phone calls, mails, etc.) and claims rate based on customers' history (in particular: claims not covered by their current covering).

2.2.2 Step B: Feature engineering

Feature engineering allows us to build relevant features based on existing variables from the initial dataset. It could be an aggregation of several features, or a transformation from numeric to categorical feature. This step is in general based on knowledge of datasets and on intuition supported by experts from specific fields about what could be the most explanatory features. We introduce some notations related to the dataset which is the output of this step.

Notation 3 We consider:

1. x the dataset obtained at the end of step B.
2. F the number of features in dataset x, i.e. the number of columns of x.
3. x_i the vector containing the value of features for the customer u_i:

$$x = (x_i)_{i \in [\![1;N]\!]}, x_i \in \mathbb{R}^F. \tag{11}$$

Then x is of dimensions N rows (i.e. customers) and F columns (i.e. features). Let us also mention that there exists automated methods to perform feature engineering such as deep feature synthesis (see [7]), but results were inconclusive in our study, due to the specificity of used datasets which requiredto aggregate them manually.
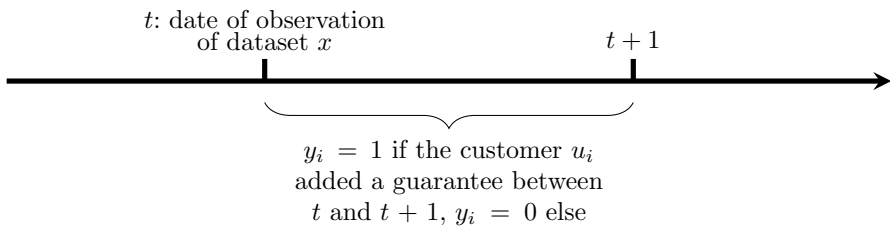
2.2.3 Step C1

Step C1 takes as an input the customers' dataset x, defined by (11). It outputs for each customer u_i, his estimated probability to add a guarantee p_t^i, defined by Eq. (9). This probability is estimated by supervised learning using a label that represents whether a customer added a guarantee in the past or not, 1 year after the extraction date of features. Learning is performed on a training dataset, which is a random subset of the rows of x. Back-testing is performed on a validation dataset, which is made of the part from x unused for training.

Notation 4 We consider:

1. y the target feature:

$$y = (y_i)_{i \in [\![1;N]\!]}, \tag{12}$$

where $y_i \in \{0, 1\}$ is the label value for the customer i. The scheme below illustrates the construction of y.

t: date of observation
of dataset x $t + 1$

$y_i = 1$ if the customer u_i
added a guarantee between
t and $t + 1$, $y_i = 0$ else

2. $\omega \in [0, 1]$ the percentage of dataset x used for training.

The mean of **y** in our dataset is 0.15, meaning that about 15% of customers added a guarantee to their car insurance product.

Remark 1 In a first approach, the learning process is made on customers who added a guarantee either by themselves or by implicit recommendations from prior contacts between agents and customers, and not necessarily on customers who improved their cover as a direct consequence of an explicit recommendation. The first reason for this is that there is no historical data on such previous recommendation campaign: it is the first time that Foyer tries this kind of automatized up-selling method. The second reason is that this approach seems to be the best alternative. Even if there is a priori no added value to suggest a guarantee to a customer who could have added the guarantee voluntarily, this method allows us to learn the profile of the customers who would be less reluctant to add a guarantee: we believe that a customer who is very hesitant to improve his cover by himself would not listen to his agent if he suggested him to do so. But we should keep in mind that the crucial point of a recommendation is the receptiveness of a customer to his agent and his propositions, which is hard to evaluate without data about human interactions with the customer.

Step C1 answers the question: to whom should we address the recommendations in priority? After testing several approaches, this step is performed by the XGBoost algorithm, based on Gradient Boosting method.

Gradient Boosting is a sequential ensemble method, first proposed by Breiman and developed by Friedman [5]. The principle of boosting is to combine weak learners (e.g. decision trees for Gradient Tree Boosting) trained in sequence to build a strong learner. In Gradient Tree Boosting, each decision tree attempts to correct errors made by the previous tree. At each iteration, a decision tree is fitted to residual error. The following algorithm presents the generic Gradient Tree Boosting method. To simplify the writing of the algorithm, we assume that we attribute numbers 1 to $\lfloor \omega N \rfloor$ to customers in the training dataset.

Algorithm 1 Friedman *Gradient Tree Boosting*

1: **Inputs:** dataset $x = (x_i)_{i \in [\![1;N]\!]}$ with N observations and F features, target feature $\boldsymbol{y} = (y_i)_{i \in [\![1;N]\!]}$, loss function Ψ, number of iterations B
2: **Result:** Vector $\hat{\boldsymbol{p}}_t$, estimation of $\boldsymbol{p}_t$, the probability of adding a guarantee
3: Extract a training dataset from x of size $\lfloor \omega N \rfloor \times F$
4: Initialize $\hat{\boldsymbol{p}}_t$: $\forall i \in [\![1; \lfloor \omega N \rfloor]\!], \hat{p}_t^i = \underset{\rho}{\text{argmin}} \sum_{j=1}^{\lfloor \omega N \rfloor} \Psi(y_j, \rho)$
5: **for** $m \in [\![1; B]\!]$ **do**
6: $\qquad$ Compute negative gradients: $z_i = \left[-\frac{\partial \Psi(y_i, p_t^i)}{\partial p_t^i} \right]_{p_t^i = \hat{p}_t^i}, i \in [\![1; \lfloor \omega N \rfloor]\!]$
7: $\qquad$ Train a decision tree h using the dataset $\{x_i, z_i\}_{i \in [\![1; \lfloor \omega N \rfloor]\!]}$
8: $\qquad$ Compute step size: $\rho \leftarrow \underset{\rho}{\text{argmin}} \sum_{i=1}^{\lfloor \omega N \rfloor} \Psi(y_i, \hat{p}_t^i + \rho \times h(x_i))$
9: $\qquad$ Update $\hat{p}_t^i$: $\hat{p}_t^i \leftarrow \hat{p}_t^i + \rho \times h(x_i), i \in [\![1; \lfloor \omega N \rfloor]\!]$
10: **end for**

XGBoost has in particular the following characteristics:

- Parallel learning: XGBoost uses multiple CPU cores to perform parallelization to build decision trees and reduces computation time,
- Regularization: XGBoost adds a regularization term which avoids over-fitting and then optimizes computation.

2.2.4 Step C2

Step C2 aims to predict which insurance cover/guarantee is most likely to be added, among the missing covers of the customers. This step answers the question: which additional insurance cover should we recommend? After testing several approaches, this step is performed by the Apriori algorithm.

The Apriori algorithm was introduced by Agrawal and Srikant [1], in order to find association rules in a dataset. (e.g. a collection of items bought together). The main applications of the Apriori algorithm are:

- Market basket analysis, i.e. finding which items are likely to be bought together, as developed in [1]. This technique is used by Amazon for their recommendation systems;
- Several types of medical data analysis, e.g. finding which combination of treatments and patient characteristics cause side effects. In [14], the authors show that Apriori is the best method to optimize search queries on biomedical databases, among other algorithms such as K-means or SVM (accuracy of 92% for Apriori versus 80% and 90% for K-means and SVM respectively);
- Auto-complete applications, i.e. finding the best words associated to a first sequence of words. This method is used by Google to auto-complete queries in the search engine.

Applied to our recommendation system, the Apriori algorithm could detect, from an initial set of guarantees subscribed by a customer, the guarantee most

frequently associated with this initial cover. Our statement is that this guarantee is most likely to be added by a customer and should be consequently the one to be recommended.

Notation 5 An association rule R is of the form:

$$R : R_1 = \{g_{r_1(1)}, \dots, g_{r_1(N_R)}\} \to R_2 = \{g_{r_2}\}, \tag{13}$$

where R_1 is a set of N_R guarantees, $r_1(k)$ the index of the kth guarantee of R_1 ($k \in [\![1;N_R]\!]$), R_2 a singleton of one guarantee of index r_2 which does not belong to R_1: $R_1 \bigcap R_2 = \emptyset$.

The Apriori algorithm generates every association rule $\mathcal{R}$ appearing from existing customers' covers.

Definition 1 The set of all association rules $\mathcal{R}$ from the customers' set $\mathcal{U}$ is:

$$\mathcal{R}_{\mathcal{U}} = \Big\{ R : R_1 \to R_2 \Big| \exists i \in [\![1;N]\!], R_1 \cup R_2 \subseteq f_t(u_i) \Big\}, \tag{14}$$

where R and $f_t(u_i)$ are respectively defined by Eqs. (13) and (7).

Example 1 If a customer subscribed to guarantees $\{g_1, g_2, g_3\}$, then association rules implied by this cover are $\{g_1, g_2\} \to \{g_3\}$, $\{g_1, g_3\} \to \{g_2\}$, $\{g_2, g_3\} \to \{g_1\}$, $\{g_1\} \to \{g_2\}$, $\{g_2\} \to \{g_1\}$, $\{g_1\} \to \{g_3\}$, $\{g_3\} \to \{g_1\}$, $\{g_2\} \to \{g_3\}$ and $\{g_3\} \to \{g_2\}$.

For each association rule, the support and the confidence are calculated and defined below.

Definition 2 The support S_R of a rule $R : R_1 \to R_2$ is the number of customers who subscribed to R_1 and R_2:

$$S_R = \#\Big\{ u_i \in \mathcal{U} \Big| f_t(u_i) \supseteq R_1 \cup R_2 \Big\}, \tag{15}$$

where $\#$ denotes the cardinality of a set. The confidence C_R of a rule $R : R_1 \to R_2$ is the proportion of customers who subscribed to R_1 and also subscribed to R_2:

$$C_R = \frac{\#\Big\{ u_i \in \mathcal{U} \Big| f_t(u_i) \supseteq R_1 \cup R_2 \Big\}}{\#\Big\{ u_i \in \mathcal{U} \Big| f_t(u_i) \supseteq R_1 \Big\}}, \tag{16}$$

where R, $\mathcal{U}$ and $f_t(u_i)$ are respectively defined by Eqs. (13), (2) and (7).

To define which guarantee is most likely to be added by each customer, we generate every association rule based on the Apriori algorithm. Then for each customer u_i, we keep the eligible rules defined below.

Definition 3 Eligible rules $\text{ER}(u_i)$ for a customer u_i are every association rule $R : R_1 \rightarrow R_2$ where R_1 is a subset of customer's cover and R_2 is not subscribed by u_i:

$$\text{ER}(u_i) = \{R : R_1 \rightarrow R_2 \in \mathcal{R}_{\mathcal{U}} | R_1 \subseteq f_t(u_i) \text{ and } R_2 \cap f_t(u_i) = \emptyset\}, i \in [\![1;N]\!], \tag{17}$$

where $R_{\mathcal{U}}$ and $f_t(u_i)$ are respectively defined by Eqs. (14) and (7).

Once eligible rules are filtered, we keep the association rule $R : R_1 \rightarrow R_2$ with the highest confidence, defined by (16). Therefore $R_2 = \{g_{\Phi(u_i)}\}$ is recommended:

$$\{g_{\Phi(u_i)}\} = R_2, \text{ such that } R : R_1 \rightarrow R_2 = \underset{R \in \text{ER}(u_i)}{\text{argmax}} C_R, \tag{18}$$

where $g_{\Phi(u_i)}$ and $\text{ER}(u_i)$ are respectively defined by Eqs. (8) and (17).

The entire process is synthesized in the following pseudo-code:

Algorithm 2 Step C2

1: Extract a training dataset from x of size $\lfloor \omega N \rfloor \times F$
2: Generate every association rule $\mathcal{R}_{\mathcal{U}}$ thanks to the Apriori algorithm from training set
3: **for** $i \in [\![1; \lfloor \omega N \rfloor]\!]$ **do**
4:     Compute eligible rules $\text{ER}(u_i)$ for customer u_i
5:     Compute $R = \underset{S \in \text{ER}(u_i)}{\text{argmax}} C_S$, where $R : R_1 \rightarrow R_2$ is the association rule with the
    highest confidence
6:     Recommend guarantee $\{g_{\Phi(u_i)}\} = R_2$
7: **end for**

2.2.5 Step D: Business rules

Business rules consist in adding additional deterministic rules based on agent expertise and product knowledge. Given Property 3, this step avoids computing recommendations which are not usable in practice. For instance, some customers are already covered by recommended guarantee because they subscribed to an old version of car insurance product, whose guarantees were defined differently. Some simple business rules downstream of the model take into account these particularities.

2.2.6 Step E: List of recommendations

Once we have every couple $\{g_{\Phi(u_i)}, p_t^i\}$, defined by (10), agents suggest guarantees for customers most likely to accept recommendations. Thus we sort this list of couples by decreasing p_t^i and we group them by agent (i.e. each customer has an assigned agent, so we generate a list of recommendations for each agent), to obtain the final list of recommendations.

3 Results

The following section summarizes results of the proposed recommendation system. After presenting values of algorithms' parameters, we first present back-testing results of both steps C1 and C2 on historical data. Then, we present back-testing results of cumulative effect of steps C1, C2 and D. Finally, we show results of a pilot phase made in collaboration with agents.

3.1 Parameters

The following subsection summarizes values of parameters introduced previously:

– we consider $N = 57.000$ policyholders of the car insurance product,
– there exists $M = 13$ different guarantees: $M_s = 10$ standard and $M_o = 3$ optional guarantees,
– we use $F = 165$ features for supervised learning on step C1,
– we consider a training dataset size of $\omega = 70\%$ of the entire dataset,
– the loss function Ψ is the square loss function: $\Psi(x) = \frac{x^2}{2}$, where Ψ is introduced in Algorithm 1,
– XGBoost uses $B = 1.000$ iterations.

3.2 Back-testing step C1

Since the objective is to avoid wrong suggestions as much as possible, at the risk of limiting the amount of recommendations, we evaluate this step on the customers most likely to accept an additional cover. To do so, we plot in Fig. 2 the rate of customers sorted by decreasing probability of acceptance who added at least one guarantee in the past (y-axis) versus the top $x\%$ of customers sorted by probability of acceptance (x-axis). The reference curve (in red) is the result given by a perfect model, which would rank every addition of cover on highest probabilities. We compare the model with some of other methods of supervised learning tested:

– A single CART decision tree (see [8]) in orange,
– Random Forest (see [3]), ensemble learning method which applies bootstrap aggregating to decision trees, in blue.

Let us detail an example. The point highlighted in purple says that, if we consider the 10% with the highest probabilities of adding a guarantee according to the XGBoost algorithm, 66% of these 10% indeed added a guarantee in the past.

Figure 2 shows that the XGBoost algorithm is the most accurate method, since XGBoost is the closest curve to the reference model on a major part of the top 20% of customers.
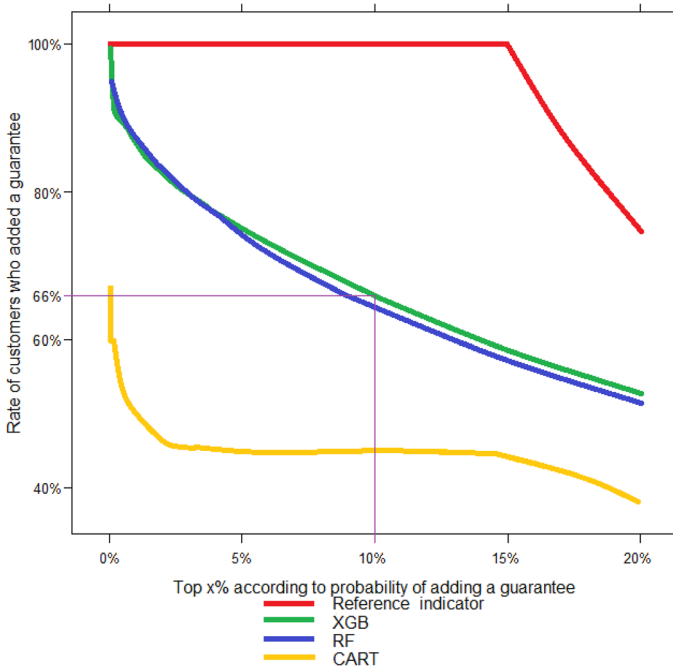
Fig. 2 Back-testing of step C1, on every customer

The profile of customers with the highest probabilities to add a guarantee is analyzed in Sect. 3.5.

3.3 Back-testing step C2

To evaluate the accuracy of this step, we consider all the customers who added a guarantee in the past, and we observe the proportion of those that added the same guarantee that would have been recommended by algorithms. We compare results of selected method, the Apriori algorithm, with other approaches for recommendation systems:

- Random: as a benchmark, we evaluate the accuracy of choosing randomly the guarantee to recommend to u_i from $f_t(u_i)^c$, where $f_t(u_i)^c$ is defined by Eq. (7).
- Popular: we recommend to u_i the most popular guarantee from $f_t(u_i)^c$, i.e. the most subscribed guarantee from $f_t(u_i)^c$ in $\mathcal{U}$, where $f_t(u_i)$ is defined by Eq. (7).
- IBCF (see [13]): this approach estimates distances between items (i.e. the guarantees) and recommends the nearest guarantee from existing customer's cover.
- UBCF (see [16]): this approach is dual with IBCF. It estimates distances between users and recommends to a customer the guarantee that nearest users, according to this distance, subscribed to.

Table 1 Step C2: comparison of methods

Method	Accuracy (%)
Random	49
Popular	67
IBCF	71
UBCF	82
SVD	72
Apriori	95

Table 2 Validation of Assumption 2

Population	Accuracy (%)
Low probabilities	93.4
High probabilities	96.6
Top 15% probabilities	97.1

– SVD (see [4]): IBCF and UBCF use distances calculated from binary user-item matrix, defined by:

$$\tilde{R} = (\tilde{R}_{i,j})_{i\in[\![1;N]\!],j\in[\![1;M]\!]} = (\mathbb{1}_{g_j\in f_t(u_i)}), \tag{19}$$

where $f_t(u_i)$ is defined by Eq. (7). SVD is a matrix factorization method which consists in decomposing R matrix described above into rectangular matrices with lower dimensions.

The results are synthesized in Table 1. They show that the Apriori approach presents the best performance on our data. Level of accuracy reached is high enough not to use more complex solutions like neural networks, which significantly increase computation time with a slight improvement of accuracy in the best case scenario, due to datasets dimensions in particular.

To validate the Assumption 2 a posteriori, we compared the accuracy of the Apriori algorithm on two sub-populations: customers with the lowest and those with the highest probabilities of adding a guarantee, calculated in step C1 (i.e. customers with a probability lower (respectively higher) than median probability on the whole population). The results are synthesized in Table 2.

The results show that for both sub-populations, the Apriori algorithm has similar accuracies, which validates (or at least does not contradict) our assumption a posteriori. The top 15% probabilities, who are the most likely to be targeted by our recommendation system and related to the 15% of customers who added a guarantee to their car insurance in the past, reach a 97.1% accuracy.
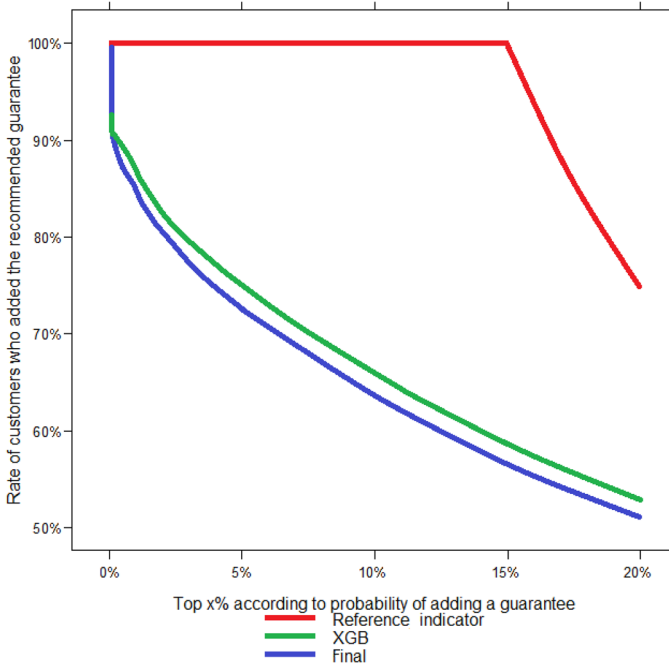
Fig. 3 Back-testing of step D, on every customer

3.4 Back-testing step D

This last back-testing phase evaluates the cumulative effect of steps C1, C2 and the deterministic rules. The result is an expected acceptance rate for recommendations, which is used as a reference for the pilot phase (see Sect. 3.5). This expected acceptance rate is calculated in a similar way that for back-testing of step C1 (see Sect. 3.2). We plot in Fig. 3 the rate of customers sorted by decreasing probability of acceptance calculated on step C1, who added the guarantee selected by step C2 and business rules (in blue), versus the top x% of customers sorted by probability of acceptance. We also plot two curves from Fig. 2: the rate obtained by the XGBoost algorithm (in green) and the reference curve given by a perfect model (in red). The difference between blue and green curves is due to customers who added a guarantee different from the one recommended.

Figure 3 shows that trend of the blue curve is in compliance with back-testing results from Sects. 3.2 and 3.3: expected acceptance rate in blue is approximately equal to 95% of the green curve. This result is in compliance with previous back-testing results: blue curve accumulates errors from both steps C1 and C2. We are able to read on the blue curve the expected acceptance rate for the recommendation system, as a function of the percentage of customers considered. For instance, if we make a recommendation to the top 10% customers according to likelihood of

adding a guarantee, we expect there will be 65% of these customers who accept their recommendation.

3.5 Pilot phase

The proposed recommendation system has been tested in a pilot phase, with the participation of four Foyer agencies. Among the portfolio of these agencies, we extracted the customers in the top 10% of the estimated probabilities of accepting a recommendation. Then the four agents proposed a recommendation to this selection of around 150 customers, by mail, phone or an appointment.

3.5.1 Implementation experience with agents

The four agents who took part in the pilot phase were selected thanks to a large portfolio and a strong motivation to test this experimental approach. Before the campaign, a presentation allowed them to discover how the recommendation system works. During the campaign, the recommendations were transmitted through a software which is daily used by agents to manage their commercial opportunities. After a recommendation, the agent typed in this interface whether the customer accepted or not, which allowed us to get the information very easily and continuously. After the campaign, agents shared their feedback during a meeting and suggested relevant potential improvements.

3.5.2 Customers' profile

We present an overview of the type of customers selected by the recommendation system, compared with the global portfolio of the agents who took part in the pilot phase. Table 3 presents some features of these customers. This table allows us to make an archetype of customers who are more likely to add a guarantee, according to the XGBoost algorithm. For instance, the first row means that the selected customers were 2.2% younger than the average customer of the four agencies' portfolio.

We could particularly note that selected customers have less guarantees than average, more products and more vehicles subscribed at Foyer, more expensive and more recent cars. These observations make sense: the XGBoost algorithm targets customers who have a reduced cover compared to their current car and their purchasing power. Besides, we could notice that selected customers have a better scoring and a higher number of amendments on their contracts: it shows that our recommendation system targets customers of better quality and those who decided to modify their contract in the past.

3.5.3 Results

We analyse the results of the pilot phase globally, by agent, and by guarantee.

Table 3 Profile of customers selected for the pilot phase

Characteristic	Delta (%)
Age	−2.2%
Living in Luxembourg City	+8.1%
Number of guarantees	−4.7%
Car insurance premium	+15.1%
Number of products	+27.4%
Insurance premium	+12.9%
Number of vehicles	+10.1%
Age of vehicles	−6.4%
Price of vehicles	+33.5%
Scoring	+0.5 level
Number of amendments	11.1%

Table 4 Pilot phase: acceptance rate by agent

Agent	Expected acceptance rate (%)	Actual acceptance rate (%)
Agent 1	61	39
Agent 2	58	48
Agent 3	60	39
Agent 4	52	21
Overall	**57**	**38**

Globally Overall acceptance rate is 38%. It is below expectations from back-testing, as shown in Table 4 (see below). This could be explained by the fact that back-testing is made on past guarantees additions, instead of past recommendations, as discussed in Sect. 2.2.3.

However this result remains promising since benchmark acceptance rate for such marketing campaigns is about 15% (see Remark 2 below for more details). This standard rate comes from previous results of a similar test based on the recommendation system developed in [10].

Remark 2 It is worth mentioning that the acceptance rates from our study and from [10] are both from a selection of the global portfolio of customers. Our acceptance rate is calculated on the top 10% recommendations according to the XGBoost algorithm, among the portfolio of the four agents. In [10], 366.998 recommendations are calculated; 737 received agent action (0.02%) and 104 were accepted (14% of the recommendations managed by agents).

Another benchmark could be classic up-selling campaigns from Foyer, which have a conversion rate from 5 to 10%. [10] also mentions that the standard industry conversion baseline is 12%. Thus, specific targeting allowed agents to increase
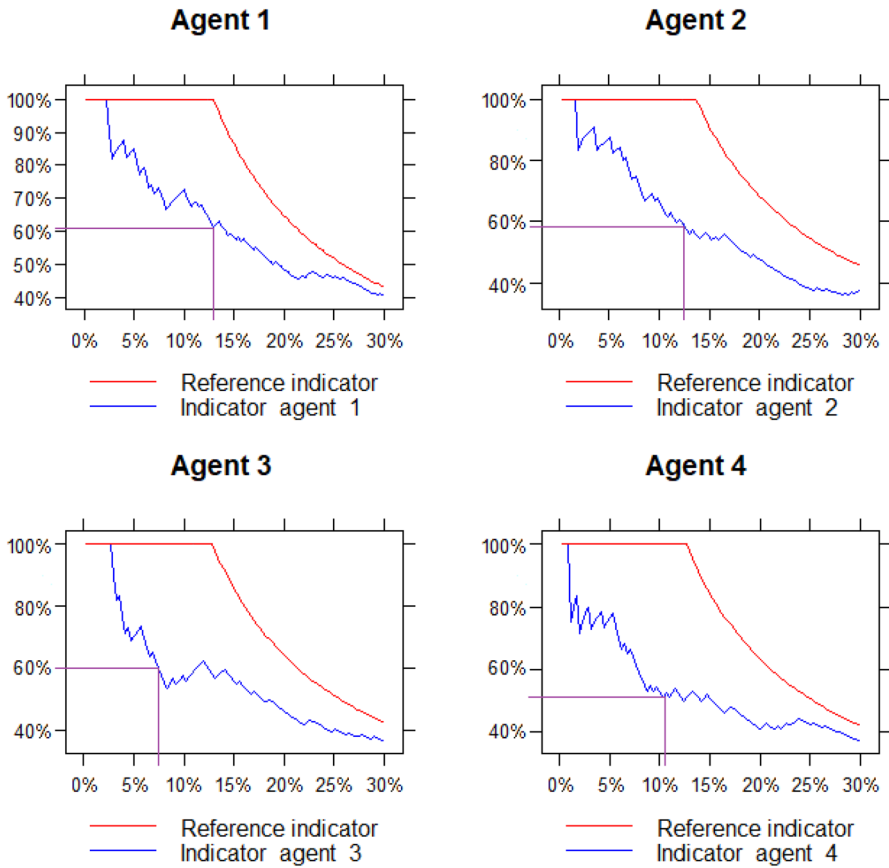
Fig. 4 Back-testing of step C1, on agents participating to pilot phase

significantly accuracy of their up-selling actions, even if acceptance rate is lower than back-testing results.

By agent Table 4 shows expected acceptance rate from back-testing and actual acceptance rate by agent, observed during one year. Expected rate acceptance is extracted from Fig. 4, which shows back-testing results from process described on Sect. 3.4 on customers from the four participating agents. On the four plots, we highlight the percentage of customers from agents' portfolios asked for a recommendation and the corresponding expected conversion rate (purple lines in Fig. 4).

As mentioned for global results, acceptance rates are below expectations from back-testing results for every agent. However, these back-testing results allowed us to detect that Agent 4 should have a lower acceptance rate before the pilot phase.

By guarantee Table 5 presents the distribution of guarantees recommended and the acceptance rates associated.

This table reveals that driver injury guarantee is by far the most recommended and the most accepted by customers as well. However, the less commonly recommended

Table 5 Pilot phase: distribution and acceptance rate of guarantees recommended

Guarantee	Percentage of recommendations suggesting the guarantee (%)	Acceptance rate (%)
Driver injury	58	47
Glass damage	24	23
Legal cover	12	33
Road accident	6	26
Overall	**100**	**38**

guarantees have a promising acceptance rate (not below 23%, which is higher that our 15% benchmark), which shows that the recommendation system could be efficient for different types of needs from customers.

3.5.4 Agents' feedback

The main reasons why some customers did not accept their recommendations, according to agents' feedback, are the following:

– Customers only subscribed to essential guarantees and do not want to spend more money on their car insurance,
– Customers already subscribed to the same type of recommended guarantee in another company. For instance, some customers already have legal cover from their employer.

Some recommendation refusals could also have been avoided because agents already suggested the guarantee to customers in the near past, without any mention about this exchange in Foyer's datasets. This inconvenience will be rectified subsequently. Besides, agents suggested that associating an explanation to every commercial opportunity would improve the recommendation system.

Remark 3 During the pilot phase, we observed that the acceptance rate decayed through time. It is explained by the fact that agents dealt first with recommendations for which they were pretty sure that they will get a positive answer, due to their knowledge of their customers. Given this observation, we could think that the recommendation system is useless since agents already knew the main part of successful recommendations. But agents highlighted the fact that one of the advantages of the recommendation system is to analyse all their portfolio equally, which allowed them to remind some customers that they would not have dealt with at the precise moment of the campaign. They also reported that some recommendation were surprisingly accepted against their intuition.

4 Future work

The following section presents improvements planned for the recommendation system, given the current results and feedback from agents.

Future work should include improvements of the proposed recommendation system:

- Explainability: the most frequent request from agents' feedbacks is to improve explainability of recommendations, i.e. why a customer receives a recommendation from the system. Some methods compute feature importance in a model or influence of a feature in a single prediction, such as SHAP analysis (see [9]).
- Integration of new relevant features: adding new features could enhance significantly the predictions. For instance, some information about contacts between customers and agents could be recovered in unexploited datasets, which would avoid to suggest a cover already recommended in the past.
- Spread to other products: we intend to generalize the recommendation system to other non-life insurance products, such that home insurance. The same architecture could be suitable but we should adapt the features and check that back-testing show the same results that car insurance.
- Specific work on life events prediction. When a life event occurs to a customer, it sometimes means that this customer has to adjust his cover. For instance, when a customer moves house, he has to adapt his home insurance policy (new address or new guarantees suited to his new house). For now, the recommendation system only takes into account vehicle changes. Thus, by forecasting these events, the recommendation system could be more accurate. Moreover, the recommendation system should be able to propose the right guarantee as soon as an event occurs.
- Challenge the Assumption 2 by testing models which make the guarantee recommended and the probability to accept the guarantee dependant. The results of the pilot phase by guarantee detailed in Sect. 3 could suggest that working on this dependency could improve the model.

References

1. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 20th international conference on very large data bases, VLDB '94 (September 1994), pp 487–499

2. Breese JS, Heckerman D, Kadie CM (2013) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the fourteenth annual conference on uncertainty in artificial intelligence, pp 43–52 (1998)
3. Breiman L (2001) Random forests. Mach Learn 45(5–32):2001
4. Cremonesi P, Koren Y, Turrin R (2010) Performance of recommender algorithms on Top-N recommendation tasks. In: Proceedings of the ACM conference on recommender systems, RecSys 2010, Barcelona, Spain, 26–30 September 2010, pp 39–46
5. Friedman (1999) Greedy function approximation: a gradient boosting machine. Ann Stat 29(5):1189–1232
6. Gomez-Uribe CA, Hunt N (2015) The Netflix recommender system: algorithms, business value, and innovation. ACM Trans Manag Inf Syst 6:13:1–13:19
7. Kanter JM, Veeramachaneni K (2015) Deep feature synthesis: towards automating data science endeavors. In: 2015 IEEE international conference on data science and advanced analytics (DSAA)
8. Loh W-Y (2011) Classification and regression trees, WIREs Data Mining and Knowledge Discovery
9. Lundberg SM, Lee S-I (2017) A unified approach to interpreting model predictions. In: Advances in neural information processing systems, 30 (NIPS 2017)
10. Qazi M, Fung G, Meissner KJ, Fontes ER (2017) An insurance recommendation system using Bayesian networks. In: Proceedings of the Eleventh ACM conference on recommender systems, August 2017, pp 274–278
11. Ramos J (2003) Using TF-IDF to determine word relevance in document queries
12. Rokach L, Shani G, Shapira B, Chapnik E, Siboni G (2013) Recommending insurance riders. ACM SAC
13. Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web. pp 285–295
14. Tummapudi TK, Uma M (2015) Effective navigation of query results using apriori algorithm. Int J Comput Sci Inf Technol 6(2):1952–1955
15. Zhang S, Yao L, Sun A, Tay Y (2018) Deep learning based recommender system: a survey and new perspectives. ACM Comput Surv 1(1):35
16. Zhao Z-D, Shang M-S (2010) User-based collaborative-filtering recommendation algorithms on Hadoop. In: 2010 third international conference on knowledge discovery and data mining, pp 478–481
17. Zibriczky D (2016) Recommender systems meet finance: a literature review. FINREC, New York