A Recommendation System For Car Insurance

Laurent Lesage, Madalina Deaconu, Antoine Lejay, Jorge Augusto Meira, Geoffrey Nichil, Radu State

A Recommendation System For Car Insurance

Laurent Lesage[†‡§], Madalina Deaconu[‡], Antoine Lejay[‡], Jorge Augusto Meira[†], Geoffrey Nichil[§], Radu State[†]

† University of Luxembourg, JFK Building, L-1855 Luxembourg
‡ Universite de Lorraine, CNRS, Inria, IECL, F-54000 Nancy
§ Foyer Assurances, 12, Rue Leon Laval, L-3372 Leudelange

December 20, 2019

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 insurance context. While traditional recommendation systems, designed for online platforms (e.g. e-commerce, videos), are constructed on huge data-sets and aim to suggest the next best offer, insurance products have specific properties which imply to adopt a different approach. Our recommendation system combines XGBoost algorithm and Apriori algorithm to choose which customer should be recommended and which cover to recommend, respectively. It has been tested in a pilot phase of hundreds of recommendations, which shows that the approach outperforms standard results for similar up-selling campaigns.

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 allowing to learn up-to-date advanced patterns from customers' behaviour and consequently target customers more accurately. In the context of recommendation system, the use of such algorithms generates commercial opportunities in function of customers and products targeted.

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 [16]). 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 [7]). It consists in formulating recommendations by filtering information from many viewpoints or data sources. The first subset of collaborative filtering techniques is 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 [17]), or items that are the most similar to items user A has already subscribed to (Item-Based Collaborative Filtering: IBCF, see [11]). The second subset of collaborative filtering techniques is model-based approach, with data mining or machine learning algorithms. A classic model is 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 [12]).

Second type of recommendation system. The second category of recommendation systems is 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 [18]). Term Frequency counts the number of times a term occurs, while Inverse Document Frequency

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

measures how rare a term is and how much information a term provides.

Third type of recommendation system. The third category is 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 allow to deal with massive information and unstructured data. The survey in [6] 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 singularities (see also [15]).
Data dimensions: the number of covers is limited to a dozen of guarantees. In comparison with thousands of books or movies proposed by online platforms, dimensions of the problem are reduced.
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 from customers in their insurance company.
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 (age, no-claims bonus level, vehicle characteristics, etc.).
Work related to insurance recommendation systems. There are few papers about insurance recommendation systems. In [2], authors present a system built for agents to recommend any type of insurance (life, umbrella, auto, etc.) based on bayesian network. The results of the pilot phase based on the recommendation system from [2] will be used as a baseline for our system. Moreover, properties and singularities of a recommendation system for insurance are listed in [15], which shows the efficiency of a recommendation system for call-centers. A survey of recommendation systems applied to finance and insurance is proposed in [14]. 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 classic models,

3. present a recommendation system whose results, on a pilot phase including hundreds of recommendations, are above standard rates of acceptance.

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

Plan. The remainder of this paper is organized as follows. We propose a suitable approach in Section 2, in accordance with the three properties described previously. Then we present results of back-testing and a pilot phase in Section 3 and conclude on future work on the recommendation system in Section 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. This product is characterized by a set of guarantees, with a particular structure. In this sense, the customers must select standard guarantees (at least one of them: third party liability), and could add optional guarantees. 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 total 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:
$$M = M_s + M_o. \tag{2.1}$$

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

$$\mathcal{U} = \{u_1, u_2, ..., u_N\}. \tag{2.2}$$

4. $\mathcal{G}$ the set of M available guarantees:

$$\mathcal{G} = \{g_1, ..., g_M\}, \tag{2.3}$$

where $g_1, ..., g_M$ 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:

$$\mathcal{G} = \mathcal{G}_s \sqcup \mathcal{G}_o, \tag{2.4}$$

$$\mathcal{G}_s = \{g_1, ..., g_{M_s}\}, \tag{2.5}$$

$$\mathcal{G}_o = \{g_{M_s+1}, ..., g_{M_s+M_o}\}, \tag{2.6}$$

where $\sqcup$ is the symbol for disjoint union.

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:

$$f_t : \left| \begin{array}{ccc} \mathcal{U} & \longrightarrow & \mathcal{P}(\mathcal{G}) \\ u_i & \longmapsto & f_t(u_i), \end{array} \right. \tag{2.7}$$

where $\mathcal{U}$ and $\mathcal{G}$ are defined by equations (2.2) and (2.3) 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 $f_t(u_i)^{\complement}$, the subset of guarantees u_i did not subscribe to, i.e.: $f_t(u_i) \sqcup f_t(u_i)^{\complement} = \mathcal{G}$.

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:

$$\Phi : \left| \begin{array}{ccc} \mathcal{U} & \longrightarrow & [\![1; M]\!] \\ u_i & \longmapsto & \Phi(u_i), \end{array} \right. \tag{2.8}$$

where $\mathcal{U}$ is defined by equation (2.2). Then $g_{\Phi(u_i)} \in f_t(u_i)^{\complement}$, i.e. the guarantee recommended does not belong to the current customer's cover.

Given Property 1, 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 one 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 1.

Notation 2. We consider:

$$\boldsymbol{p_t} = (p_t^i)_{i\in[\![1;N]\!]} = \left(\mathbb{P}\Big[g_{\Phi(u_i)} \in f_{t+1}(u_i) | f_t(u_i) \cap g_{\Phi(u_i)} = \emptyset\Big]\right)_{i\in[\![1;N]\!]}, \quad (2.9)$$

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 equation (2.7).

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

$$\{g_{\Phi(u_i)}, p_t^i\}, i \in [\![1; N]\!], \quad (2.10)$$

where $g_{\Phi(u_i)}$ and p_t^i are defined by equations (2.8) and (2.9) respectively.

2.2 The model

We propose the following approach to build the targeted recommendation system, illustrated by Figure 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 equations (2.9) and (2.8) 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 1). Indeed, since we focus on a small amount of covers (Property 1), Apriori algorithm (see Section 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 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 Sections 3.2 and 3.3, respectively).

After steps C1 and C2, business rules are applied to tackle Property 1. Then the recommendation system outputs the final list of recommendations, ordered by decreasing probability of adding a cover. The following subsections describe each step separately.

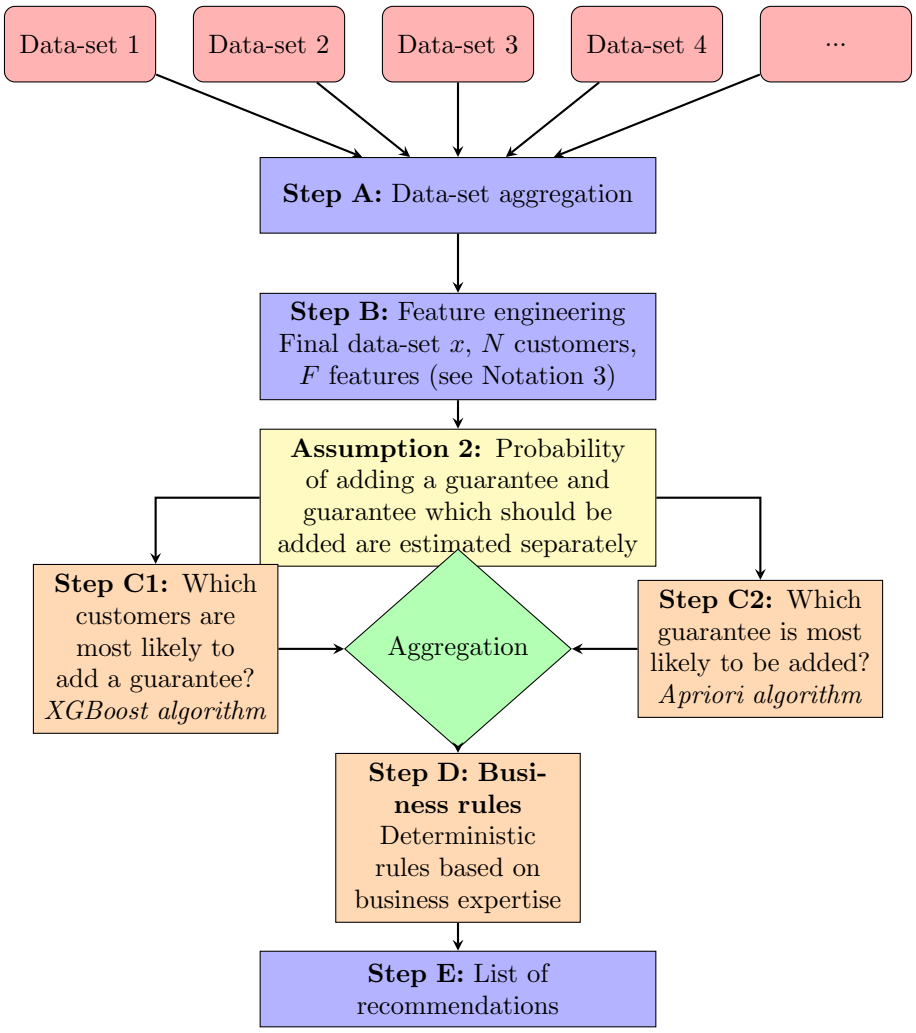


Figure 1: Global architecture of the recommendation system

2.2.1 Step A: Data-set aggregation

We built an unique data-set from multiple internal data sources. This data-set 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 to build relevant features based on existing variables from the initial data-set. 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 data-sets and on intuition supported by experts from specific fields about what could be the most explanatory features. We introduce some notations related to the data-set which is the output of this step.

Notation 3. We consider:

1. x the data-set obtained at the end of step B.
2. F the number of features in data-set x, i.e. the number of columns of x.
3. $\boldsymbol{x_i}$ the vector containing the value of features for the customer u_i:

$$x = (\boldsymbol{x_i})_{i \in [\![1;N]\!]}, \boldsymbol{x_i} \in \mathbb{R}^F. \tag{2.11}$$

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 [5]), but results were non conclusive in our study, due to the specificity of used data-sets which required to aggregate them manually.

2.2.3 Step C1

Step C1 takes as an input the customers' data-set x, defined by (2.11). It outputs for each customer u_i, his estimated probability to add a guarantee p_t^i, defined by equation (2.9). This probability is estimated by supervised learning using a label that represents whether a customer added a guarantee in the past or not, one year after the extraction date of features. Learning is performed on training data-set, which is a random extraction from x. Back-testing is performed on validation data-set, which is made of the part from x unused for training.

Notation 4. We consider:

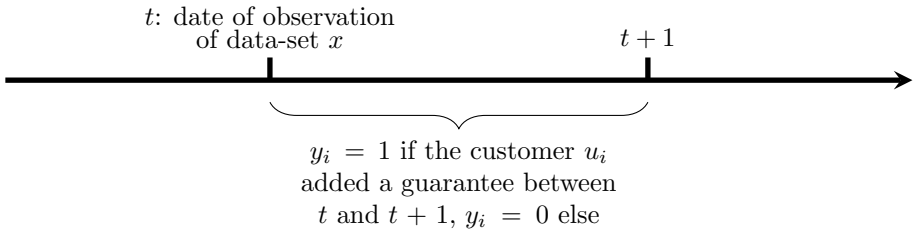
1. $\boldsymbol{y}$ the target feature:

$$\boldsymbol{y} = (y_i)_{i \in [\![1;N]\!]}, \tag{2.12}$$

where $y_i \in \{0, 1\}$ is the label value for the customer i.

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

The scheme below illustrates the construction of $\boldsymbol{y}$.



Mean of $\boldsymbol{y}$ is 0.15, meaning that about 15% of customers added a guarantee to their car insurance product.

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

Gradient Boosting is a sequential ensemble method, first proposed by Breiman and developed by Friedman in [9]. The principle of boosting is to combine weak learners (e.g. decision trees for Gradient Boosting) trained in sequence to build a strong learner. In Gradient 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 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 data-set.

Algorithm 1 Friedman *Gradient Boosting*

Inputs: Data-set $x = (x_i)_{i \in [\![1;N]\!]}$ with N observations and F features, target feature $\boldsymbol{y} = (y_i)_{i \in [\![1;N]\!]}$, loss function Ψ, number of iterations B
Result: Vector $\hat{\boldsymbol{p_t}}$, estimation of $\boldsymbol{p_t}$, the probability of adding a guarantee
Extract a training data-set from x of size $\lfloor \omega N \rfloor \times F$
Initialize $\hat{\boldsymbol{p_t}}$: $\forall i \in [\![1; \lfloor \omega N \rfloor]\!], \hat{p}_t^i = \underset{\rho}{\operatorname{argmin}} \sum_{j=1}^{\lfloor \omega N \rfloor} \Psi(y_j, \rho)$
for $m \in [\![1; B]\!]$ **do**
    Compute negative gradients: $z_i = -\frac{\partial \Psi(y_i, p_t^i)}{\partial p_t^i}\Big|_{p_t^i = \hat{p}_t^i}$, $i \in [\![1; \lfloor \omega N \rfloor]\!]$
    Train a decision tree h using the data-set $\{x_i, z_i\}_{i \in [\![1; \lfloor \omega N \rfloor]\!]}$
    Compute step size: $\rho \leftarrow \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{\lfloor \omega N \rfloor} \Psi(y_i, \hat{p}_t^i + \rho \times h(x_i))$
    Update $\hat{p}_t^i$: $\hat{p}_t^i \leftarrow \hat{p}_t^i + \rho \times h(x_i)$, $i \in [\![1; \lfloor \omega N \rfloor]\!]$
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 allows to avoid over-fitting and then optimize computation.

2.2.4 Step C2

Step C2 aims to predict which insurance cover/guarantee is most likely to be added, among the whole 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 Apriori algorithm.

The Apriori algorithm was introduced by Agrawal and Srikant in [8], in order to find association rules in a data-set (e.g. a collection of items bought together, as developed in [8]). For the recommendation system, it allows us to detect from any customer's existing cover the guarantee most frequently assigned with this initial set of guarantees. 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:

$$R : R_1 = \{g_{r_1(1)}, ..., g_{r_1(N_R)}\} \rightarrow R_2 = \{g_{r_2}\}, \tag{2.13}$$

where R_1 is a set of N_R guarantees, $r_1(k)$ the index of the k^{th} 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:

$$\mathcal{R}_{\mathcal{U}} = \Big\{ R : R_1 \rightarrow R_2 \Big| \exists i \in [\![1; N]\!], \{R_1 \cup R_2\} \subseteq f_t(u_i) \Big\}, \tag{2.14}$$

where R and $f_t(u_i)$ are respectively defined by equations (2.13) and (2.7).

If a customer subscribed to guarantees $\{g_1, g_2, g_3\}$, then association rules implied by this cover are $\{g_1, g_2\} \rightarrow \{g_3\}$, $\{g_1, g_3\} \rightarrow \{g_2\}$, $\{g_2, g_3\} \rightarrow \{g_1\}$, $\{g_1\} \rightarrow \{g_2\}$, $\{g_2\} \rightarrow \{g_1\}$, $\{g_1\} \rightarrow \{g_3\}$, $\{g_3\} \rightarrow \{g_1\}$, $\{g_2\} \rightarrow \{g_3\}$ and $\{g_3\} \rightarrow \{g_2\}$. For each association rule, the support and the confidence are calculated and defined below.

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

$$S_R = \#\Big\{ u_i \in \mathcal{U} \Big| f_t(u_i) \supseteq \{R_1 \cup R_2\} \Big\}, \tag{2.15}$$

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

$$C_R = \frac{\#\Big\{u_i \in \mathcal{U}\Big| f_t(u_i) \supseteq \{R_1 \cup R_2\}\Big\}}{\#\Big\{u_i \in \mathcal{U}\Big| f_t(u_i) \supseteq R_1\Big\}}, \tag{2.16}$$

where R, $\mathcal{U}$ and $f_t(u_i)$ are respectively defined by equations (2.13), (2.2) and (2.7).

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.

Definition 3. Eligible rules $\mathrm{ER}(u_i)$ 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:

$$\mathrm{ER}(u_i) = \{R : R_1 \rightarrow R_2 \in \mathcal{R}_{\mathcal{U}} \Big| R_1 \subseteq f_t(u_i) \text{ and } R_2 \cap f_t(u_i) = \emptyset\}, i \in [\![1; N]\!], \tag{2.17}$$

where $R_{\mathcal{U}}$ and $f_t(u_i)$ are respectively defined by equations (2.14) and (2.7).

Once eligible rules are filtered, we keep the association rule $R : R_1 \rightarrow R_2$ with the highest confidence, defined by (2.16). Therefore $R_2 = \{g_{\Phi(u_i)}\}$ is recommended:

$$\{g_{\Phi(u_i)}\} = R_2, \text{ such that } R : R_1 \rightarrow R_2 = \underset{R \in \mathrm{ER}(u_i)}{\operatorname{argmax}} C_R, \tag{2.18}$$

where $g_{\Phi(u_i)}$ and $\mathrm{ER}(u_i)$ are respectively defined by equations (2.8) and (2.17).

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

Algorithm 2 Step C2

1: Extract a training data-set from x of size $\lfloor \omega N \rfloor \times F$
2: Generate every association rule $\mathcal{R}_{\mathcal{U}}$ thanks to Apriori algorithm from training set
3: **for** $i \in [\![1; \lfloor \omega N \rfloor]\!]$ **do**
4: Compute eligible rules $\mathrm{ER}(u_i)$ for customer u_i
5: Compute $R = \underset{S \in \mathrm{ER}(u_i)}{\operatorname{argmax}} C_S$, 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 1, this step allows to avoid 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 allow to take into account these singularities.

2.2.6 Step E: List of recommendations

Once we have every couple $\{g_{\Phi(u_i)}, p_t^i\}$, defined by (2.10), 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, 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 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 data-set size of $\omega = 70\%$ of the entire data-set,
- The loss function Ψ is the square loss function: $\Psi(x) = \frac{x^2}{2}$, where Ψ is introduced in Algorithm 1,
- XGBoost algorithm 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 Figure 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 [4]) in orange,
- Random Forest (see [13]), ensemble learning method which applies bootstrap aggregating to decision trees, in blue.

Figure 2 shows results of this back-testing. It proves that 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.

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, 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 $f_t(u_i)^{\complement}$, where $f_t(u_i)^{\complement}$ is defined by equation (2.7),
- Popular: we recommend to u_i the most popular guarantee from $f_t(u_i)^{\complement}$, i.e. the most subscribed guarantee from $f_t(u_i)^{\complement}$ in $\mathcal{U}$, where $f_t(u_i)$ is defined by equation (2.7).
- IBCF (see [11]): this approach estimates distances between items (i.e. the guarantees) and recommends the nearest guarantee from existing customer's cover.
- UBCF (see [17]): 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.
- SVD (see [12]): IBCF and UBCF use distances calculated from binary user-item matrix, defined by:

$$\tilde{R} = (\tilde{R}_{i,j})_{i\in[\![1;N]\!],j\in[\![1;M]\!]} = (\mathbb{1}_{g_j\in f_t(u_i)}), \tag{3.1}$$

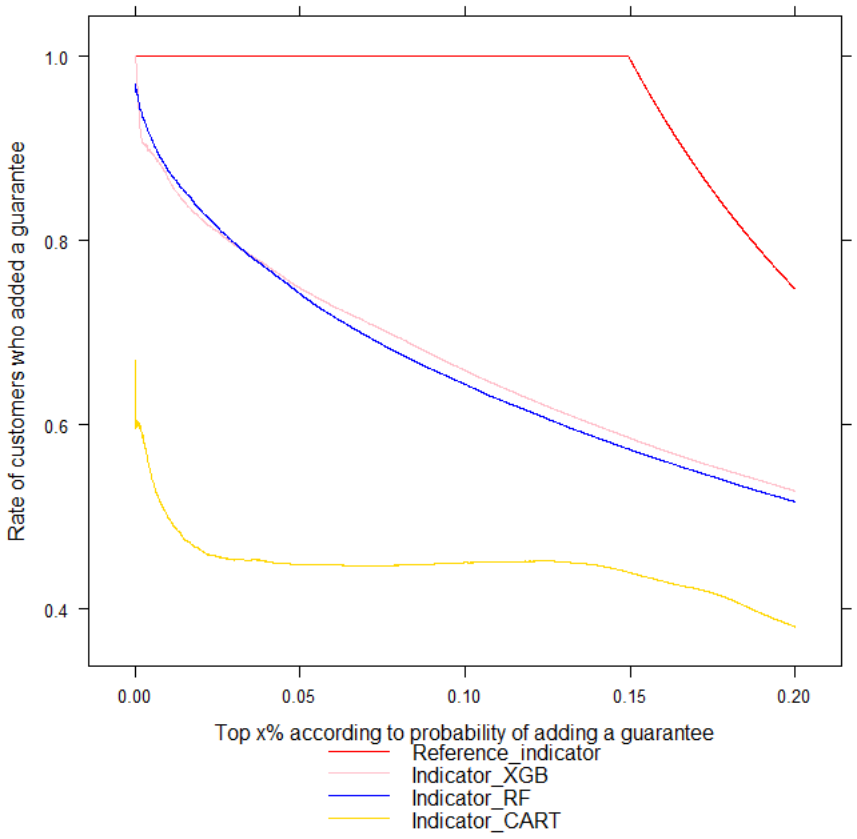


Figure 2: Back-testing of step C1, on every customer

where $f_t(u_i)$ is defined by equation (2.7). 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 our 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 data-sets dimensions in particular.

Table 1: Step C2 - comparison of methods

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

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 Section 3.5). This expected acceptance rate is calculated in a similar way that for back-testing of step C1 (see Section 3.2). We plot in Figure 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 Figure 2: the rate obtained by XGBoost algorithm (in pink) and the reference curve given by a perfect model (in red). The difference between blue and pink 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 Sections 3.2 and 3.3: expected acceptance rate in blue is approximately equal to 95% of the pink 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.

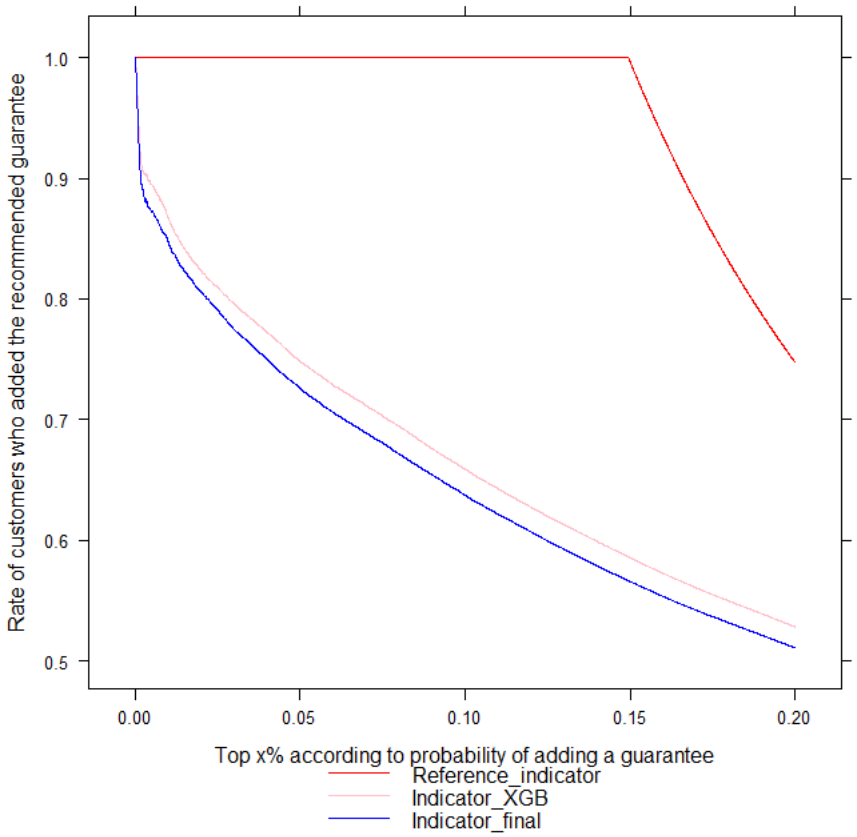


Figure 3: Back-testing of step D, on every customer

3.5 Pilot phase

The proposed recommendation system has been tested in a pilot phase. Hundreds of recommendations were calculated and then proposed to customers by four agents, by mail, phone or real talk. Customers chosen for this campaign are those with the highest estimated probability of accepting a recommendation.

Table 2 shows expected rate acceptance from back-testing and actual acceptance rate by agent. Expected rate acceptance is extracted from Figure 4, which shows back-testing results from process described on Section 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 (green lines in Figure 4).

Table 2: 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** %

Overall acceptance rate is 38%. It is below expectations from back-testing, as shown in Table 2. This could be explained by the fact that back-testing is made on past guarantees additions, instead of past recommendations. The algorithm learns on customers who added a guarantee by themselves and not on customers who accepted a recommendation. Yet there is a difference between adding a guarantee spontaneously and accepting to add one, based on a recommendation. Past guarantees additions is an overestimation of what could be accepted from recommendations.

However this result remains promising since benchmark acceptance rate for such marketing campaigns is about 15%. This standard rate comes from previous results of a similar test based on the recommendation system developed in [2]. Moreover, classic up-selling campaigns from Foyer have a conversion rate from 5% to 10%. Specific targeting allowed agents to increase significantly accuracy of their up-selling actions, even if acceptance rate is lower than back-testing results.

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

- Customers only subscribed to essential guarantees and do not want to spend more money on their car insurance,

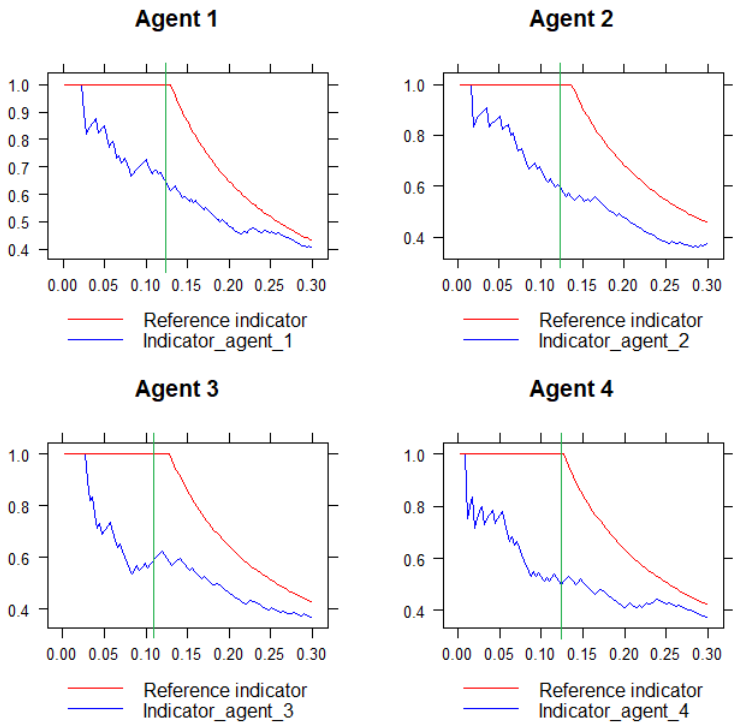


Figure 4: Back-testing of step C1, on agents participating to pilot phase

- 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, with any mention about this exchange in main data-sets. This inconvenience will be rectified subsequently.

4 Future work

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

Future work should include improvements of the proposed recommendation system:

- Explainability: the most frequent request from agents' feed-backs is to improve explainability of recommendations, i.e. why a customer receives a recommendation from the system. Some methods allow to compute feature importance in a model or influence of a feature in a single prediction, such as SHAP analysis (see [1]).

- 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 data-sets, 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). Thus, by detecting this events, the recommendation system could be more accurate.

We developed this work in a strong collaboration with Foyer Assurances, leader of individual and professional insurance in Luxembourg, which provided the domain specific knowledge and use cases. We also gratefully acknowledge the funding received towards our project (number 13659700) from the Luxembourgish National Research Fund (FNR).

References

[1] Scott M. Lundberg and Su-In Lee, A Unified Approach to Interpreting Model Predictions. 2017. Advances in Neural Information Processing Systems 30 (NIPS 2017).

[2] Qazi, Fung, Meissner and Fontes, An Insurance Recommendation System Using Bayesian Networks. 2017. RecSys'17 (August 2017).

[3] Chao-Ying Joanne Peng, Kuk Lida Lee, Gary M. Ingersoll, An Introduction to Logistic Regression Analysis and Reporting, 2002. The Journal of Educational Research.

[4] Wei-Yin Loh, Classification and regression trees, 2011. WIREs Data Mining and Knowledge Discovery.

[5] James Max Kanter and Kalyan Veeramachaneni, Deep Feature Synthesis: Towards Automating Data Science Endeavors, 2015. 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA).

[6] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2018. Deep Learning based Recommender System: A Survey and New Perspectives. ACM Comput. Surv. 1, 1, Article 1 (July 2018), 35 pages.

[7] John S. Breese, David Heckerman and Carl Myers Kadie, Empirical Analysis of Predictive Algorithms for Collaborative Filtering, 2013. CoRR (Computing Research Repository), 2013.

[8] Agrawal and Srikant, Fast Algorithms for Mining Association Rules. 1994. Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pages 487-499 (September 1994).

[9] Friedman, Greedy Function Approximation: A Gradient Boosting Machine. 1999. IMS 1999 Reitz Lecture.

[10] Laub, Taimre, Pollett, Hawkes Processes. 2015. https://arxiv.org/abs/1507.02822.

[11] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, Item-Based Collaborative Filtering Recommendation Algorithms. 2001. Proceedings of the 10th international conference on World Wide Web, Pages 285-295.

[12] Paolo Cremonesi, Yehuda Koren and Roberto Turrin, Performance of Recommender Algorithms on Top-N Recommendation Tasks. 2010. RecSys'10.

[13] Leo Breiman, Random Forests, 2001. Machine Learning, 45, 5–32, 2001.

[14] David Zibriczky, Recommender Systems meet Finance: A literature review. FINREC 2016.

[15] Lior Rokach, Guy Shani, Bracha Shapira, Eyal Chapnik, Gali Siboni, Recommending insurance riders. 2013. ACM SAC, 2013.

[16] Carlos A. Gomez-Uribe and Neil Hunt. 2015. The Netflix recommender system: Algorithms, business value, and innovation. ACM Trans. Manage. Inf. Syst. 6, 4, Article 13 (December 2015), 19 pages.

[17] Zhi-Dan Zhao and Ming-Sheng Shang, User-based Collaborative-Filtering Recommendation Algorithms on Hadoop. 2010. 2010 Third International Conference on Knowledge Discovery and Data Mining.

[18] Juan Ramos, Using TF-IDF to Determine Word Relevance in Document Queries, 2003.

[19] Tianqi Chen and Carlos Guestrin, XGBoost: A Scalable Tree Boosting System. KDD '16 Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016), Pages 785-794.