Behavioral patterns of long term saving : Predictive analysis of adverse behaviors on a savings portfolio

Babacar SOW, Executive Assistant to the CAO of CNP Assurances
Thomas BEHAR, Chief Actuary Officer (CAO) of CNP Assurances
Wenyao JIN, datascientist at CNP Assurances

November 8, 2017

Abstract

The switch of insured people toward unit linked (UL) contracts is more and more a strategic orientation for life insurers. Unlike Euro saving contracts (Euro) where a surrender value is constantly guaranteed, UL contracts don't include a guarantee on a capital implying a reduction of the solvency II capital requirement. Clients bear the whole financial risk. Regardless of the tax advantages influencing both the individual and legal entity clients, the low interest rate environment and the probable volatility underlying on equities due to the uncertainty of the geopolitical environment could lead part of the savings portfolio to see Euro contracts as a safe investment. In addition, the occurrence of the lapse risk in the event of a sudden rise of interest rates could involve the shift of customers towards new profitable investments. In the French life insurance market, the insured people have the right to withdraw at any time all - *total surrender* - or part - *partial surrender* - of their savings at the surrender value. A transfer could also be requested from Euro funds toward UL underlying funds - *this action is called Arbitrage*.

To face these behavioral risks with financial origin and consequences, usual leverages used by traditional insurers are often limited to descriptive statistics. In addition, remediation actions are restricted on deterministic rules linked to the experience : For example, a usual recall for active clients to maintain their activity on the contract. These rules are not always appropriate for new business contracts and passive clients since no record on their activity do exist. Then, it could be useful for insurers to explore predictive methods in order to anticipate future actions of insured people and take steps to limit the implied consequences on portfolio's profitability. In addition, the complexity and overlaps between behaviors' main features - *Identity, Terms of contracts, Transactional trends, Macro economy, Geo-localization* - make the use of machine learning approaches particularly designed for. Life insurers could get into a preventive position by optimizing their commercial retention offers and heading a more suitable risk management strategy.

This article aims to present and structure an adaptation of data science algorithms on a savings portfolio in the construction and the analysis of a predictive model applied on different management acts of a saving contract - *Adverse arbitrages and Surrenders*. These management acts are predicted separately in order to avoid wrong readings involved by crossed actions. Besides, the remedial measures for these two actions are suggested to be different. It introduces in a second time the sequence mining of these different events through association rules. Sequential pattern mining techniques are broadly used in retailing industry to help the arrangement of supermarket's shelves. It is also used in bio informatics for the study of nucleotides sequences within a DNA molecule. The related methodology described in this article behaves by analogy and gives a probabilistic vision on the occurrence of a given adverse management act regarding an experienced sequence at a given past period.

The document starts with defining an appropriate approach for feature engineering purpose on a savings portfolio, including a particular treatment for some category of features, leveling of time series data issue, features selection and so on. This step also enables the problem definition by introducing the specification of the target. Once roll up to a classification problem, the treatment is based on an adaptation of different classical category of machine learning algorithms - *Ensemble methods and Linear Regression* - where parameters are assessed regarding the LIFT and the AUC - *Area under Curve* - metrics. The AUC metric is chosen because of its insensitivity to the level of the target - *An evolution of the target's level each year doesn't affect the generalization of the model* - while the LIFT metric measures the performance of the prediction regarding a perfect random model.

Last, the results from the predictive model are exposed as well as some perspectives. We consider a real savings portfolio of about 50 000 clients where the average distribution over the last 6 months for the surrender management act (for amounts over than 40 000 €) is about 4% - *that is to say about 2 000 clients made a > 40 000€ withdraw on their savings during the last 6 months.* We show that the predictive model over perform a usual market model based on the practice of targeting on clients who ever realize adverse actions on the considered period - *we call it the empirical model.* By sorting through the 10% best scores, the predictive model gives a 40% recall versus 15% for an empirical model and 4% for a full random model, in line with the average distribution - *the recall metric (applied over the 10% selection) measures the proportion of true positives selected by the predictive model with respect to the whole relevant clients that really made an action on the considered period.* By evidence the results are much better if we consider more than 10% best scores for the recall metric and less than 10% best scores for the precision of predictions performed. This model could be used to supply a decision-making tool based on an objectification of opportunity gains in launching special offers.

Key words : Predictive analysis, Machine learning, Insured behavior, Saving contracts

1 Introduction

The unfavorable European economic conditions with low interest rates combined to geopolitical uncertainty lead life insurers to **move their strategy toward operational efficiency**. At the same time, in many industries, clients are more and more challenging on their expectations and demanding on tailor made services. That is the case in the whole insurance sector. In this context, we consider a set of clients' behavioral actions - *subscription cancellation, churn or lapse and arbitrages from Euro to Unit Linked (UL) funds* - on a life insurer savings portfolio. These actions are considered as adverse regarding performance and risk indicators. Consequently, the ability to predict and anticipate them seems to be a practical way to generate knowledge and wealth on the portfolio as a whole. All comments here arise from techniques used on different industries, assuming that using analogies from other fields is a good composition. This article deals with a use of machine learning and data mining approaches to help on this direction. The objective is to share a practical approach (tested on a real savings portfolio) for colleagues and peers to find out a value added between the translation of much theoretical papers and much operational one. In fact, as practitioners, we remark that, as well-filled they could be, literatures on these fields are often either inaccessible by their complexity or overly superficial to have a solid understanding on what is happening. Advantages for life insurers could be expressed on different ways :

- **Less adverse lapses on the portfolio since the average cost of retention is often significantly less than acquisition cost of new clients**. Like telecoms industry, traditional insurers could encounter a complex disruption of clients' behaviors with a significant ability to move from a provider to another one. Main determinants of these behaviors are in general across broad cross-dimensional effects. Even if the experienced churn rate for insurers are rarely at an alarming level, the upgrade of marketing and commercial methods with predictive analysis approaches could be a good lever for anticipation
- **More accuracy on underwriting risk and capital management by orienting customers toward risk-less products** (here UL products). Especially for European life insurance companies under Solvency II, capital optimization through an adaptation of the product mix becomes a key point on the risk management framework *(the lapse risk represents about 61% of the life underwriting SCR under solvency 2, according to the French regulator in [ACP15], published from the French market).*

Usual commercial models already vote in UL or limiting lapses and adverse arbitrages through different ways : commercial offers in a limited period to boost sales and giving the right to a subsidized guarantee conditional to a minimum investment on UL compartment, limitation of guaranteed rates on Euro saving funds, promotion of management mandate to avoid massive arbitrages from UL funds, better pedagogy on UL-based products, Commission plan for sales staffs conditional to UL volume on sales etc. Whatever the potential lever to activate, a such action could not be realized on the whole client portfolio by evidence, for cost effective concerns. Therefore, the key success factor to really activate a benefit remains on answering the following question :

On which customer could we apply this action for a better response? If the definition of considered levers to apply, as described above, is specific to each life insurer (from its customer experience understanding), the answer of this question could always be caught through a machine learning based approach.

In this paper, we propose a set of "recipes" for supervised learning methods applied on a life insurers' portfolio. Many constraints could appear, due to their database structure underlying the long term risk they hold. As recalled above, literatures in this field offer a wealth of information and some techniques related in this document are quite abundant. However, life actuaries may have difficulties to find out a way to realize in concrete terms such a study on their portfolio. We **first describe a process to build a practical way to get into a classification problem** from classic data of a life insurer (Identity, Terms of contracts, Transactional trends, Macro economy, Geo-localization). Ensemble methods (random forest, boosting) are particularly designed for giving a generalizable and robust prediction to this problem. Beyond methods of fine tuning Model performances, conclusions from experienced sequences of different management acts (**Given an observation period, is there any link between successive management acts** - *let's say the sequence [Partial Surrender (PS) / Arbitrage Euro -> UL (Arb Euro -> UL)]* - **and a Total Surrender (TS) occurred at a later time ?**) is presented afterwards. The sequence mining problem is caught from **the GSP-algorithm** mentioned in [RA95]. We propose an adaptation of this algorithm to our portfolio and we show that the introduction of the sequence mining features into the predictive algorithm also improve the predictive model performance as viewed by AUC and LIFT metrics described in [TS15]. We **next present an adaptation of staking methods to boost classifiers' performances** starting from [SD04] where the domination of combining classifiers (stacking methods) in comparison to selecting the best one is broadly discussed. **Last, the results from a such model applied to a real 50 000 clients portfolio** are exposed as well as some perspectives of improvement.

2 Problem definition : Get into a Classification Problem

2.1 Understanding of the supervised learning problem

We first recall the distinction between unsupervised learning problems where the goal is to model the underlying structure of an input database - *Example : association rules that gives a scoring to make a particular action conditionally to previous distinctive features or looking for patterns in the Data by clustering* - and supervised learning problems. With the supervised learning problem, the input variables $\{x_1, ..., x_d\}$ and the output variable $\{y\}$ are given and a defined algorithm is applied to learn the mapping function f(.) from the inputs to the output. This category of learning problem is further grouped into regression and classification problems : the first one where the y variable is a category - *let's say $\{OK$ -> 1 ; KO -> $0\}$* - is called a Classification problem and the other one where the y variable is continue and belongs to R is a Regression problem - *Example : the amount of mathematical reserve* -
In all cases we can write the relationship between $\hat{y}$, the estimated y and the observations $\tilde{x}$ of x :

$$\hat{y} \simeq \hat{f}(\tilde{x})$$

Let's consider the following settings and write mathematically some key points of the supervised classification problem we focused on :

- Data $x_i \in \chi = R^d$, $y_i \in \gamma = \{1, ..., k\}$ for $i = 1, ...n$
- x_i is an input and y_i is an output (both already observed)
- x_i are called **Features** and $x_i \in \chi$
- y_i is called **Label** : $y_i \in \gamma$

The goal is to predict the label y_j associated to each new observation x_j, given the so called **training database** $\{x_i, y_i\}$. We can already notice by the way some feelings about these definitions : a large d combined to a small n relates to computational biology-like concerns - *Example : very long DNA sequences but few observations from patients* - Marketing issues are generally represented by

small d and large n - *many clients and relative few features describing the information held on them*. Problems with large n and large d could be flagged as *Big data* concerns.

What to do ? Minimize the function R_n wrt to the **prediction function** $f : R^d \longrightarrow R$

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, f(x_i))$$

Where :

- l is called the **loss function**. $l(y_i, f(x_i))$ small means y_i is close to $f(x_i)$.
- $R_n(f)$ is called **empirical risk**. The quality of the prediction function is measured by its empirical risk. The best prediction function $\hat{f}$ is the one that minimize $R_n(f)$.
  $\hat{f} \in \{argmin_{\{f\}}\ R_n(f)\}$
- A learning algorithm is a function $Algo$ that links any training database - *that means recorded observations* $\{x_i, y_i\}$ - to the best estimation $\hat{f}$ of the prediction function.
  $Algo : \{x_i, y_i\} \longmapsto \hat{f}$.
- The computation of $\hat{f}$ is called **training**

Actuaries already handle statistics. One can see that the above objective - predicting the unknown information - is shared with the statistics field. However, we can notice that any "A priori" assumption is necessary and all reflexions will focus on building and fitting an algorithm on data observations no matter how the real distribution probability is known. So, in machine learning, generalization of the solution made is tackled through - *get a new dataset and apply the fitted model on that* - where in Statistics it's considered a "super population" inference - *the given dataset is a sample from a larger population that we are really interested in and we thing on that way by taking some statistical assumptions and somehow connect the sample dataset to that larger population* . Case in general, it cannot be considered all the eventual estimators for f function. [TH08] recalls that minimizing $R_n(f)$ leads to infinitely many solutions : any function $\hat{f}$ passing through the training points $\{x_i, y_i\}$ is a solution and any particular solution chosen might be a poor predictor at test points $\{x_j, y_j\}$ different from the training points. Then **choose a class of algorithm comes in practice to choose a restricted classes of $\hat{f}$ functions and resolve the above optimization problem with an underlying loss function** l. The three main steps - *as a construct of the mind* - to resolve a supervised learning problem could then be restricted on (1) the choice of a class of prediction function f, (2) the choice of a loss function l that assesses the proximity of the model predictions from the real experienced data and (3) the resolution of the optimization problem to deliver the best parameters of the fitted model.

As this paper is practical oriented, we make on section C of the appendices of this paper, a description of what would be the above tree steps for two classical classes of algorithms broadly used - the *Linear Regression and the Decision Tree*.

2.2 The problem definition

2.2.1 Definition of an adverse behavior

A customer behavior is branded adverse if it implies a negative impact on key performance and risk indicators. Therefore we consider adverse arbitrages and lapse risk in our savings portfolio. The action of Arbitrage is totally free and we distinguish between a transfer from the Euro fund and a transfer from the UL funds. The last one (Arb UL -> Euro) is seen as adverse from the life insurer point of view since it conducts to less expected value. We also focus on surrender risk as a whole (total and partial lapses) and assume that it is an adverse action on the portfolio despite the fact that a lapse from loss-making contracts may be favorable.

2.2.2 Definition of the Target

As we can learn in [sld16], we are reduced to build a matrix $\{x_i \ ; \ y_i\}$ and solve the supervised learning problem. The real performance of the prediction is known at the end of the P_2 time period. At a calculation date T_0, the predictive model have to be applied on a structured database built from P_1 period observations. Figure 1 shows a representation of the target $Y_{unknown}$ with respect to the input data $X = \{x_i \ \}$. Each line of the matrix represents a client and its different features are arranged by columns. In our applied case, we fix the target period P_2 as to be 6 months. Case in general, this window could be calibrated to be chosen as optimal with respect to the prediction scores. Therefore, some operational constraints to implement real retention actions after the run of the predictive model could avoid a shorter observation period. As recalled on the introduction of this paper, one of the major issues of the machine learning problem is the fact that the training database has to be built. So to build the training dataset, we scan the database of clients and we tag into the y_i variable respectively "1" and "0" for clients for whom we notice at least one adverse action and no adverse action during [T-6 ; T0] time period.

Keep in mind that the construction of the matrix X can be tricky because many features appears as time series at the valuation date. In other words, the whole information for a given client into the P_1 time period have to hold in one line of the training dataset. This point will be treated on section 3.

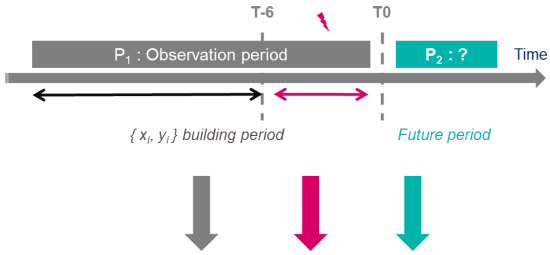


Figure 1: Construction of the target

$X = \{x_1, x_2, ..., x_d \ \}$	$\{y_i\}$	$Y_{unknown}$
Data from client 1	1	1
Data from client 2	0	0
...	...	...
Data from client n	1	1

Table 1: y=1 if an adverse action occurs (unknown at time T_0) on Period P_2, otherwise 0

2.3 Data sources and representations

Machine learning tools are rarely used on savings portfolios because the appreciation of the risk level is usually based on a few set of variables (Age, Gender, Guaranteed rate, Seniority). Yet, data is often enough present on databases.

2.3.1 Data representations

We can basically distinguish quantitative from qualitative data. Quantitative data are values describing a measurable quantity. Computational and comparison operations can be performed on it. It includes continuous features (financial market indexes, mathematical reserves, funds daily quotations) and discrete one (age, seniority). Qualitative data shows characteristics (nature of guarantee, sex and social category). Even if with numerous appearance, the corresponding values are taken as modalities (zip code, level of guaranteed rate). Order relation can exist on these features (results of an opinion poll - from not satisfied to fully satisfied).

As usual packages on machine learning tools available in open source uses numerical data, a transcription from raw data to numerical data is often necessary.

2.3.2 Data sources

Data providing is a big issue in its own right. The onerousness of underlying problems to ensure data consistency or to make it easily usable could be quickly sensitive. Let's take missing data issues : it can be linked either to IT treatments failure or to an additional information on the database structure - For example, an intentional absence of some confidential data. Treatments in missing data operated in most of the case are quite generic for machines learning projects : a full suppression of missing data if the proportion is quite very small or the replacement by null for numerical missing data.
Either way, data could be extracted from two main sources: from internal or external.

2.3.3 Internal Data

Insurers often structure their database by silo. It's quite not frequent to find a container with the whole information concerning the client. Then we can find on a life insurer database different typology of data :

- identity data : Name, address, social category, marital situation, gender ...
- data linked to the contracts and the products : subscription date, status of the contract, date of exit - *function of the status of the contract* , guaranteed rates ..
- historic of past mathematical reserves
- transactional data : activity of the insurance policies in time

Fine analysis of log files for insurers with Internet distribution platform could also be useful.

2.3.4 External Data

In addition, public data is also available for free (statistics on social and fiscal data per zip code from public institutions' publications, financial market data). The web could also be a serious provider but need some particular skills on web scraping methods as mentioned in [CF15].

3 The overall process of the machine learning exploration on life insurance databases

The main purpose of this step is to get a large X/Y matrix enriched by new features specially designed for improving the validation performance. We illustrate in appendix C, from the Linear Regression Problem, the way that feature engineering process allows to capture non linearity effects.

3.1 Data Preprocessing

In this kind of study, raw data are acquired without any transformation. These data are characterized by the presence of missing and inconsistent data. Their treatment, which is often called cleansing, consists of filtering or reconstituting them by hand. An expertise on the data domain and a particular vigilance is required to carry out a complete treatment that is reliable. To do this, some statistics to detect inconsistencies could help.

3.2 Feature Engineering

To put the data in standard form X (Explanatory variables) / Y (Target), and then feed it to a classifier machine learning, we have to apply aggregation approaches. This structure can also be considered as a first step of feature engineering in the sense that we must choose the attributes to aggregate and the methods of aggregation to favor an information rather than another.

$X = \{id, name, date, management\ act..., x_d\ \}$	$\{y_i\}$	$Y_{unknown}$
0001, Dufour, 2015-05-30, Lapse	1	1
0001, Dufour, 2015-10-10, Free premium (Prime Libre)	0	0
...	...	...
0001, Dufour, 2015-12-20, Lapse	1	1

Table 2: How do this information hold in one single line for the client 0001?

3.2.1 Time series treatment

Let's take an illustration from our database. The client - *Client i* - performed 3 actions in the observation period - *called P_1 in section 2*.

To aggregate these transactional data, the approach applied is the following :

- Define a period of time - *6 months in this paper*- (to not confused with the length of P_2 time period).
- Apply an aggregation method for all client actions included in the period P_1 - *see RFM model* [LIU05] - **sum** of amount flow, **frequency** of actions identified, **most** recent action taken
- Transform the result obtained in column with a tag for each period

With the Figure 2 we show an illustration of applying the above 3 steps. $Q1_{2009}$, ..$Q1_2010$ tags are created to aggregate the former representation of the database - *highlighted in green* - by sum.

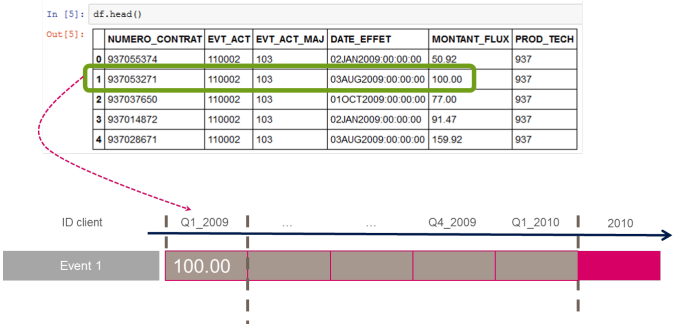


Figure 2: Construction of the X matrix

We notice that the choice of the 6-month period in defining the target may allow capturing seasonality effects, which is very classical on observed phenomena. A quick analysis with descriptive statistics makes it possible to ascertain this point. The risk of this aggregation method is that the aggregation period is too long and involves loss of causality information. Sensitivity to the size of the window adopted could be measured for more comfortability.

A way to capture financial market indexes - *or any macro economic index* - is to create features about the behavior of customers on a given time expressed from the macro economic index's dynamic.

Over P_1 observation period, we create 10 new features representing the 10 deciles $d_1, d_2, ..., d_{10}$. How was the macro economic index - *with 12 months memory* - when the client made an adverse action? We feel that by answering that question, the underlying features created could be powerful. The X dataset is scanned and for each customer, the idea is to find the points where an adverse action is made - *represented by circles on Figure 3*. We read the last 12 months from that point and record the related decile with respect to the evolution of that macro economic index. Several aggregation methods could be applied if many actions are recorded (the RFM model principles could be used : recency, frequency and monetary for aggregation purpose).

3.2.2 Aggregating by Attributes

Aside from transactional data, there are other "one to many" relationships. For example, a client may have multiple contracts registered with the company. To aggregate these relations, the general

Figure 3: Construction of new features from a macro economic index

approach is as follows:

1. Choose the most important attributes for the prediction model
2. Apply an aggregation method - *max, min, mean for cash attributes or a counter for categorical attributes*- by using RFM aggregates
3. Transform the result obtained in column with a tag for the attribute.

Loss of information is almost inevitable in some cases. Let's reuse a client-contract example: if the set of contracts is classified into 50 categories on an attribute. 50 columns should be produced after applying the above procedure. The X table size created would be virtually unmanageable. A possible treatment for this type of problem is to group the categories that have a small population

3.2.3 Enrichment of Data

This step often called feature engineering consists in making the right indicators to differentiate the two target populations - *the "0" from the "1"* through the explanatory variables. These indicators will then be added to table X to facilitate learning to classify. In this part, a typical approach is to list all the indicators that are likely to be useful, and then use a search algorithm to find the best combination of indicators. This classical approach is too greedy according to time. It will not be applied in the illustrations we make in this paper. In fact, the variables are numerous, which entails a gigantic size to be sought. Our indicators are found by model performance analysis, statistical studies, and empirical knowledge about the portfolio.

3.3 Algorithms

Several machine learning methodologies are applied in the context of this study. We present here the principle of these models and the reason why we apply them. Some examples are given to illustrate the effect of these methods. Without burdening the explanation, they all carry the - Arb Euro -> UL - event as target. The counterpart on surrenders is analogous.

3.3.1 Random Forest

Among all discriminatory classifiers, Random Forest (RF) is today a powerful, robust and fast model. It is an ensemble model based on the vote of several Decision Trees on samples drawn randomly according to the lines - *called "Tree Bagging"* and according to the columns - *known as "Feature Sampling"*. The first notion is to reduce the variance of the set - *an average of K random variables iid has a variance 1 / K times reduced.* Excluding the hypothesis of independence, the latter notion is likely to reduce the correlation between the different features. The description of the algorithm can be found in Appendix A.

Random Forest has non-exclusive advantages over other algorithms:

- **Quick Calibration**
  Based on the decision tree algorithm which is a greedy algorithm, random forest inherits this property and remains a fast model. The complexity created by the bagging step is compensated in the production context by the fact that the shafts can be driven in parallel.

- **Mix of quantitative and categorical variables**
  As the decision tree, the RF processes one variable per iteration. The quantitative and categorical variables can mix and undergo the same treatment. This gives a great advantage to the context of business projects.

- **Robustness to scale changes, renormalization, etc.**

- **The impurity calculation used by the decision tree is invariant with respect to the variable scale**

- **Taking Missing Data into Account**
  Decision tree algorithms are robust with data in the presence of missing data

- **Easy interpretation.**
  Random Forest automatically gives, after its training, an importance of variables *An example is given in Figure 4* calculated according to their importance to help the model to classify the population. **This output is also an indicator for good data enrichment.**

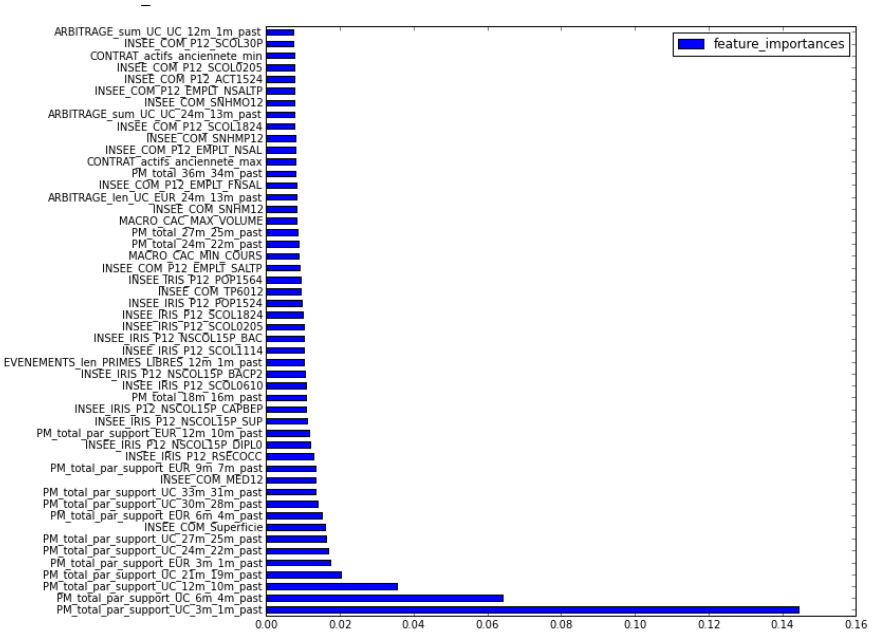


Figure 4: Feature Importance given by a Random Forest

3.3.2 XG Boost

Proposed by [FRI01b] in 2001 (then not so recent), gradient boosting designates the greedy algorithm which uses an additive strategy to aggregate several trees and drives them by a gradient criterion. At each iteration step, the loss function gradient for the current model is computed and a new regression tree is calculated by taking the gradient as a new target. The tree is then added to the current model by minimizing the loss function. The description of the algorithm can be found in Appendix B.

The comparison of this model with random forest is difficult because they share all the advantages given by the Decision Tree. A disadvantage for gradient boosting is that more parameters are requested compared to random forest. The fact of being iterative but not parallel also poses more temporal complexity. In our application, we used the algorithm of extreme gradient boosting, which is slightly modified boosting gradient for a multi-kernel implementation. One advantage of Gradient Boosting is that it is more robust on highly biased data.

These two models gives already pretty good results on targeting - *see section 5.* In the following, two additional techniques enhancing the performance on predictions will be presented : the sequence mining and the stacking method. A first step is to introduce the valuation metric under which the models are assessed.

4 Modeling

After the data processing steps that occupy most of the time of the machine learning project, the classifiers can begin to learn about these data. Articles about machine learning offer an algorithm box with various empirical characteristics and performances. **One of the highest lever for actuaries from now is the easy access to fully implemented packages on Python - *see [sld16]*.**

We have applied most of the algorithms that are adapted to the problem and then compare their performance.

A major difficulty encountered in this step for savings portfolios is related to the sensitivity to the considered period. The economic context that seems to lead the reactions of the customers could be better taken into account in the construction of X matrix. One technique that allows to have more data lines from different time periods and better train the model - *because of the presence of different time periods and different economic contexts* - could be as follows :

1. Defining a time lapses $t_1, t_2, t_3, \ldots, t_n$, with $t_n - t_{n-1} = l$ we have chosen $l = 6\, months$ for matching with our database
2. on each t_i, use the historic of time r before, to build explicative features - *we take* $r = 3$ *years under the assumption that information before 3 years ago will not impact on the future*
3. on each t_i, Use the data of 6 months later to build the target.
4. Concatenate the X/Y matrices obtained from data $t_1, t_2, t_3, \ldots, t_n$ Constructed, without differentiating them.
5. Suppose the number of lines and the history length of the original data is n, h. The number of lines after this treatment is $(h - r)/l * n$

Figure 5 illustrates the superposition of different built datasets from different time periods as explained above. Based on this "enormous" dataset in the right configuration $\{x_i, y_i\}$ for the handling of supervised learning algorithms, a simple "shuffle method" picking randomly on lines without altering the distribution of the target, could divide the large matrix on 2 datasets : the training dataset and the validation dataset. With this approach, a customer could appear n times in the dataset. This point seems not to tarnish the final performances presented in Section 5. A deeper analysis of the theoretical formulation to valid this point could be required. We do not explore that trail in this paper.
In addition, the major fear when dealing with machine learning problems is the over-fitting. We present in section 4.2.1 a cross validation approach limiting any over-fitting at the learning step.

4.1 Metrics of Valuation

Once the model is constructed, an evaluation phase will help us to differentiate the models by their consistency with the objective of prediction. Given the commercial interest of the topic, we will want to use this model to target a defined size of customers. That leads to the choice of LIFT as the evaluation criterion. LIFT is a very used criterion in the marketing field by its intuitive meaning.

We use the "surrender" act as a target to explain this point. Figure 3 is the confusion matrix that explains the prediction states for a given client. B and C represent the two types of error in a classification problem.

From this matrix we have:

$$precision = \frac{A}{A + C}$$

$$percentage\ of\ the\ target = \frac{A + C}{A + B + C + D}$$

Status of the client	Prediction : Lapse	prediction : no lapse
Reality: lapse	A	C
Reality: no lapse	B	D

Table 3: confusion matrix

$$LIFT = \frac{precision}{percentage\ of\ the\ target}$$

$$LIFT\ 10\% = \frac{precision\ over\ 10\%\ best\ score\ of\ the\ population}{percentage\ of\ the\ target}$$

LIFT represents the precision of the model over the precision given by a random model.

However, the value of LIFT has a disadvantage: when the target percentage changes, the value of LIFT will also change and they will no longer be comparable to each other. In our project, we needed to make a comparison of model capacity in time while the target population changes in time. So we need another criterion to achieve that goal.

ROC curve is the graph plotted with true positive in y, and false positive in x :

$$True\ positives = \frac{A}{A+B},\ False\ positives = \frac{C}{C+D}$$

AUC is the area under the ROC curve. [TS15] showed that AUC does not change with the percentage of the target. AUC, which is a criterion also widely used in practice, correspond exactly to our need.

4.2 Fine Tunning the Performance of Models

4.2.1 Temporal K-fold

Since the relative complexity of models used could lead to the over-fitting phenomenon - *the parameters of the model are too specific to the training set* -, cross-validation is recognized to be a calibration procedure necessary to avoid over-fitting. The procedure involves cutting the data available for the training step in a number of ways into learning databases and validation databases. The different basic pairs thus constructed are then used to evaluate the parameters of the model. A combination of the calibrated parameters will be chosen.

Given that our data has a temporal characteristic, a special data slicing procedure is required. What is used in our case is a cutting by K-fold. The procedure is illustrated in figure 5

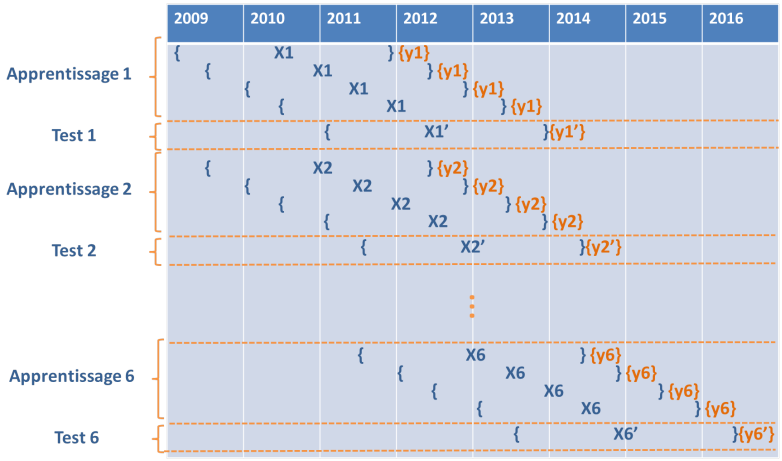


Figure 5: Temporal K-fold

4.2.2 Undersampling Method

One of the characteristics of our data is that they are biased on the target - *most clients do not take action on their contract, illustrated by Table 2*. Some algorithms, especially when we are interested in the non-zero population in relation to the target. In the two algorithms we have presented, Random forest is affected by the imbalanced data, while Gradient Boosting is not.

Period	total population	population zero(%)
2012 S1	33707	2.8%
2012 S2	34769	2.4%
2013 S1	35985	2.3%
2013 S2	37862	2.4%
2014 S1	39648	2.1%
2014 S2	44131	1.6%
2015 S1	51440	2.4%
2015 S2	58193	1.7%
2016 S1	65312	2.0%
2016 S2	72730	1.5%

Table 4: At each time step we have considered in our case study, the target is hugely biased toward the "zero population"

Undersampling and oversampling are the most applied stochastic sampling techniques to address the imbalance problem on the target. Suppose P is the so-called zero population on the target, Q is the non-zero target population. Undersampling will perform a downsampling on P, and oversampling multiplies the size of Q by the rebate draws.

[DC03] shows that undersampling is more powerful than oversampling. Yet undersampling reduces the size of the data set, which is critical for some projects where data is already scarce. In our paper, we tested several coefficients. We show in figure 6, - *with 1 represents not undersampled, 0.1 indicates that Q is reduced* 90% -

We conclude that 0.1 is the best reduction coefficient in our case study.

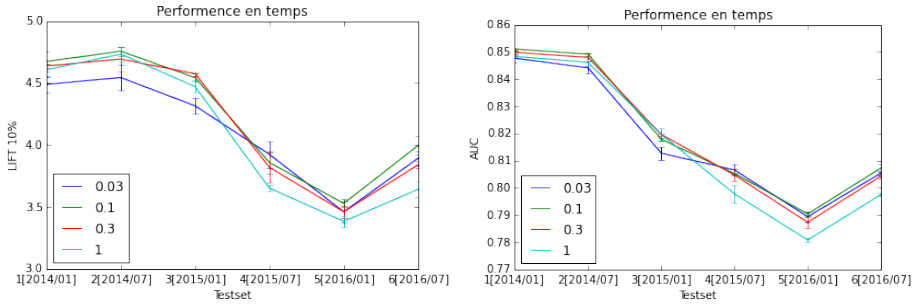


Figure 6: Performance (LIFT and AUC) of the Random Forest run over different size of "zero populations" for different time periods

4.2.3 Sequence Mining

Sequence mining is a data mining technique often used to explore sequential data. These data could be ordered by time, for example transactional data for one subscribed customer during a certain period. Other type of data can also be sequential when ordered by its physical nature, typically human genome sequence data in Bioinformatics.
Since we are dealing with transactional data, **we choose to model it as sequence of actions so as to generate features that capture the information of order between actions**. In the following sections, we present the definition of the sequence mining problem and the way we use it for generating features. Some descriptive statistics on respectively lapse and arbitrage risks are computed in figure 7. It focuses on population on which we have recorded an adverse action during the second semester of 2016. We then compute the frequencies this population had performed a management act on its insurance policy during different past time periods - *among Arbitrage from UL->Euro, Arbitrage from Euro->UL, Total Surrender and partial surrender management actions.*

The results show that if we interest in the Arb Euro -> UL action, customers that perform this action have made the other actions without any specificity. However, if we interest in the lapse risk, we notice that the past occurrence of a surrender action seems to induct a lapse action into the considered period, without a certainty. Hence, we have no idea of witch hidden subsequence is useful or not for prediction purpose. We guess that running a sequence mining algorithm into the different management acts could provide a value added on predictive models and enhance its performance results.

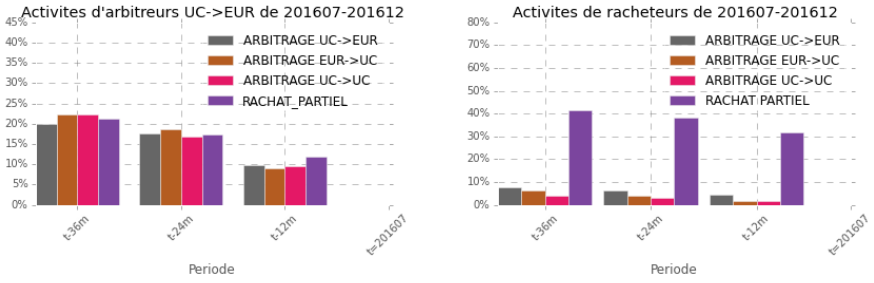


Figure 7: Representation of the past activity of customers when we focused on clients who make an adverse action during the $[2016_{07}, 2016_{12}]$ time period : Arbitrage UL->Euro (on the left) and lapse (on the right)

In addition, we complete the description of adaptations we brought into this method into the appendix D. We also describe an algorithm for frequent subsequence mining, a prediction rule generation method and a feature generation method in order to aliment our prediction model with powerful features and by the way improve the prediction performances.

Problem Definition

Let $I = \{i_1, i_2, ...i_m\}$ be a set of literals called items. An itemset X is a set of items ($X \supseteq I$). A sequence $s = \langle t_1, t_2, ..., t_n \rangle$ is an ordered set of itemsets. The length of s is defined as the number of items contained in s. We use $|s|$ to represent the length of s. A sequence of length k is called a k-sequence.

Consider two sequences $s_1 = \langle a_1, a_2, ..., a_n \rangle$ and $s_1 = \langle b_1, b_2, ..., b_n \rangle$. s_2 is a subsequence of s_1 if there exist integers $j_1, j_2, ..., j_l$, such that $1 \leq j_1 < j_2 < ... < j_l \leq n$ and $b_1 \subseteq a_{j_1}, b_2 \subseteq a_{j_2}, ..., b_n \subseteq a_{j_n}$. We represent this relationship by $s_2 \sqsubseteq s_1$.

To applicate sequence mining and its definition to real projects, items and sequences needs to be defined based on data at disposal and the information that needs to be captured. This procedure requires business knowledge on the data. In our case, we have applied two approaches to define our dictionary of items. One with a large mesh that takes a client action as one item, the other uses a finer mesh that define a cash flow on a "technical products" - *this term is used to*

call a specific fund in which the client invests - as an item - *the smallest element for life insurance contract management, corresponds to a real investment fund and represents the mesh on which flux are aggregated. Notice that the choice of the mesh is not specific to the problem and so lecturers could specify their own definition.* These two approaches will be respectively presented in this section.

Large Mesh

Client actions are naturally separated by its financial characteristics. The non-comprehensive list of these actions and their explanation is presented here. Actions will be noted by its French abbreviation throughout this paper.

- Free Premium(PL) : client puts some money into his/her contract
- Partial surrender(RP) : client withdraws a part of savings from his/her contract
- Subscription(SCP) : client subscribe to a contract
- churn(RT) : client withdraws all his/her savings and close the contract
- Arbitrage UC to EUR(AUE) : client transfers an amount of savings from unit-linked products to non-united-linked products
- Arbitrage EUR to UC(AEU) : client transfers an amount of savings from non-unit-linked products to united-linked products
- Arbitrage EUR to EUR(AEE) : client transfers an amount of savings from non-unit-linked products to other non-unit-linked products
- Arbitrage UC to UC(AUU) : client transfers an amount of savings from unit-linked products to other unit-linked products

Every client has a sequence of actions derived from his or her transactional history. We define the set of actions taking place in one day as an item set. The sequence can be generated for each client within any chosen period of time. For example, a client with a history in Table 5 will be presented as $\langle\{AUE, AUU\}, \{RP\}, \{PL\}\rangle$. We list here with several examples among its subsequences for better comprehension: $\langle\{PL\}\rangle$, $\langle\{AUE\}, \{PL\}\rangle$, $\langle\{AUU, AUE\}, \{RP\}\rangle$.

Client ID	Date	Action Type	Action Sum
C00001	2015/06/21	AUE	2000
C00001	2015/06/21	AUU	5000
C00001	2015/08/15	RP	3000
C00001	2016/01/03	PL	2000

Table 5: Example of actions

This approach and its generated subsequences are easier to understand. Items are few, so the calculation cost is not high. We see in appendix D how to construct association rules with mined subsequences. These rules can help to have a better understanding of the data and to interpret predictive models.

However there is one major drawback for this approach : actions can be overly simplified as only the financial nature of actions is captured and information about the "technical products" is thrown away. A typical concern comes from the obvious distinction between a free premium on a "Euro fund technical product" and the one on the unit-linked side. The former shows proneness to lower risk and the latter shows an appetite for higher gain. Similar attributes as the duration of the investment into the fund (could be temporal or permanent), and the choice of the management mode (free or mandated to an asset manager) can also create similar concerns.

Adding these types of information is not easy. If we separate a free premium action into free premium on Euro fund and free premium on UL funds, the total occurrence for each premium

action will be reduced. With all the other attributes added all together, this kind of separation can create a serious dilution on subsequences support and eventually unintentional prunes by minimum support threshold. According to [SRI96], this kind of problem can be solved by defining taxonomies for interrelation between actions. In our project, we tested another approach with finer mesh to address this problem.

Fine Mesh

The "technical product" is the smallest element for the management of a life insurance contract. Each "technical product" is directly linked to an invested fund while savings of each contract will be shared among these different funds. Each transactional action from the client can be interpreted by a cash flow among technical products (positive flow equals to buying or paying a premium, negative flow equals to selling or making lapses). Defining items of sequence mining with flows on "technical products" can help to model each client' action without losing any product information (product type, product duration, management mode). Taking the example from Table 5, same data with "technical product" will be represented as in Table 6. By removing the sum of flow, we define each item as a combination of flow sign and "technical product" name. Finally, we construct the sequence as $\langle\{+EurFund, +AXAEurope, -HSBCEurope\}, \{-EurFund\}, \{+AXAEurope\}\rangle$.

Client ID	Date	Technical Product	Product Type	Flow Sum
C00001	2015/06/21	Eur Fund	non-unit-linked	+2000
C00001	2015/06/21	AXA Europe	unit-linked	+5000
C00001	2015/06/21	HSBC Europe	unit-linked	-7000
C00001	2015/08/15	Eur Fund	non-unit-linked	-3000
C00001	2016/01/03	AXA Europe	unit-linked	+2000

Table 6: Example of flows on technical products

With this approach, our sequences are constructed with flows that directly demonstrate clients' appetite toward a specific product. The original action can also more or less reconstructed by combination of these items ($\{+EurFund, -AXAEurope\} = AUE$). Another great advantage of this approach is that we don't need to map our "technical products" to its attributes before defining any items. In practice, less mapping means cleaner data as tables used for mapping can come from various resources and they are often out of date or not complete.

This method has however some shortcomings. It's less easy to interpret and visualize as actions are deconstructed to "technical products". Moreover, "technical products" are numerous. In our case, more than 400 "technical products" are found, which means over than 800 type of flows - *positive flow + negative flow* are possible. This creates ultimately more subsequence candidates, thus heavier calculation cost are generated. **An intelligent algorithm is essential to make the mining possible because the number of subsequence candidates is exponential to their length**. A such algorithm (GSP Algorithm) is presented in appendix D.

Considering its advantages and drawbacks, this approach will be used to generate features for predictive models. We show in section 5 that features provided by the sequence mining process appears in the feature importance run. Moreover, we demonstrate also in section 5 that the introduction of sequence mining features lead to better results - *in terms of robustness and performance.*

4.2.4 Stacking

In the same logic to find out some levers in oder to improve predictions, here is a quite famous method widely used in machine learning competition today to enhance overall performance by combining several classifiers. The idea is to take advantage of each classifier in the final prediction.

The Framework

Stacking is concerned with combining multiple classifiers generated by using different learning algorithms $L_1, ..., L_N$ on a single dataset S, which consists of examples $s_i = (x_i, y_i)$, i.e., pairs of feature vectors (x_i) and their classifications (y_i). In the first phase, a set of base-level classifiers $C_1, ..., C_N$, where $C_i = L_i(S)$. In the second phase, a meta-level classifier is learned that combines the outputs of the base-level classifiers.

To generate a training set for learning the meta-level classifier, a leave-one-out or a cross validation procedure is applied. For leave-one-out, we apply each of the base-level learning algorithms to almost the entire dataset, leaving one example for testing:$\forall i = 1, ..., n : \forall k = 1, ..., N : C_k^i = L_k(S - s_i)$. We then use the learned classifiers to generate predictions for $si : \hat{y_i^k} = C_k^i(x_i)$. The meta-level dataset consists of examples of the form $((\hat{y_1^i}, ..., \hat{y_n^i}), y_i)$,where the features are the predictions of the base-level classifiers and the class is the correct class of the example at hand. When performing, say, 10-fold cross validation, instead of leaving out one example at a time, subsets of size one-tenth of the original dataset are left out and the predictions of the learned base-level classifiers obtained on these.

The method

Several framework in proposed through articles. [SD04] summarizes several methods for such as Multi-response linear regression(MLR), Meta Decision Tree(MDT), Multi-response model tree(MRMT) and proved that stacking with MLR using probability distribution produced by base-level classifiers outperforms other methods. Since our problem is a binary prediction, we'll restrict to use Linear Regression (LR) on outputs of a a first stage prediction to train on probability distributions.

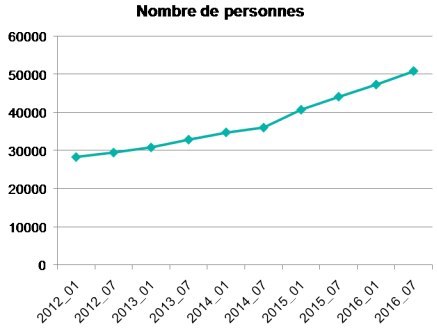
Results - *see section 5* - on this application shows that the meta model from LR systematically over performs the single machine learning algorithms. We build a practical way to optimize the use of available data for the training set. Our approach is presented in appendix E.

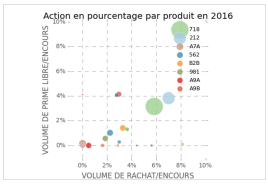
5 Results and main conclusions

This section presents successively the main results and conclusions beyond this paper. We deliberately choose to not always indicate in what risk (lapse or adverse arbitrage) the performances are computed since the goal is here to illustrate how machine learning techniques help to improve the behavioral patterns and not so an analysis focused on the portfolio we have taken to run the case study.

5.1 Oversee of the portfolio

We choose to run the above approaches in a portfolio with activity. The customer base is growing and a consequent historic is recorded.





Action \ Risque 2016	Nombre de personnes	Montant total
Rachats	9005 (19%)	55 million (5,6%)
Arbitrage	1516 (3,2%)	627 million (0.5%)

Figure 8: The dynamism of the selected portfolio is presented into the left. The activity seen both through volumes of transactions and proportions of the related population is shown into the right

5.2 The contribution of the stacking approach

The stacking method illustrated in Figure 9 over performs the classical machine learning algorithms (about 0.3 points above the boosting algorithm - xgb and 0.8 points above the GLM algorithm - sgd) with respect to the LIFT metric. Among the numerous classifiers tested, we represent here tree of them (xgb, rf and sgd) in representation of both ensemble methods and generalized linear models. The figure 9 shows the 10%LIFT metric computed in the validation set on four time periods. We can observe stability on prediction made no matter in which reference period we place on.

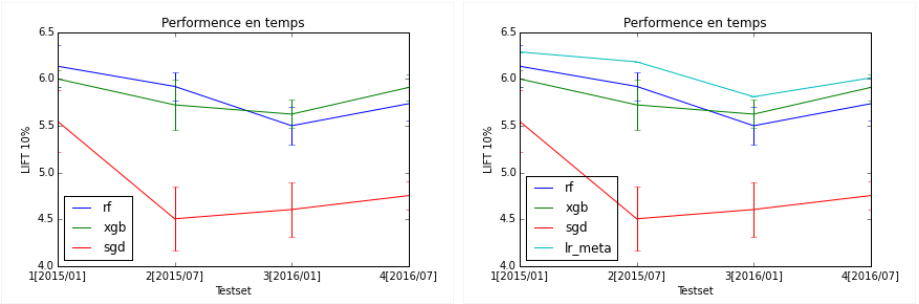
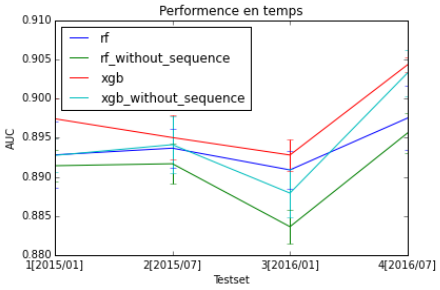


Figure 9: (Left - without running the meta model and Right - with the meta model). The meta model seems to automatically choose the best model run of the first stage. The gradient boosting algorithm, random forest and stochastic gradient descent (linear classification model) are respectively named and xgb, rf, sgd.

5.3 The contribution of the sequence mining approach

The generation of sequence mining features in addition to the RFM based features - *see section 3* - conducts to the following conclusions :

- An improvement of the AUC metric - *see figure 10* - We recall that conclusions from the AUC metric is more generalizable since LIFT metric depends on the underlying distribution of the related adverse action.
- Sequence mining features systematically appear in the 50 first "powerful" features among the whole features on the X matrix - *over than 700 features after the feature engineering step.* - *See figure 11* -



SEQUENCE_((+F18))	1.919914
SEQUENCE_((+X40))	1.635415
SEQUENCE_((+LM7))	1.619409
SEQUENCE_((+LU3))	1.546669
SEQUENCE_((+F41))	3.02708
SEQUENCE_((-F17))	1.118764
SEQUENCE_((-F17),(+F17))	3.841122
SEQUENCE_((+X40,-F17))	2.106105

Figure 10: with the addition of sequence mining features into the X database, we notice an improvement of the AUC metric both from random forest and gradient boosting algorithm (with respect to the Arb UL->Euro target). The left table shows a LIFT metric defined as the probability to have the 1 target conditionally to the occurrence of the mentioned subsequence with respect to the natural probability to get target 1 : $Computed\,LIFT = \frac{P(Y=1\,/\,occurrence of the subsequence)}{P(Y=1)}$

.

Figure 10 shows on its left side an extraction of some LIFT scores obtained on some frequent subsequences found by the sequence mining algorithm. For example it appears that the subsequence $SEQUENCE_(+X40, -F17))$ (where the F17 funds represent the Euro fund) appears with a LIFT of 2.1. That means the appearance of this sequence could be associated to better target the 1 label (2.1 times more than a random approach).

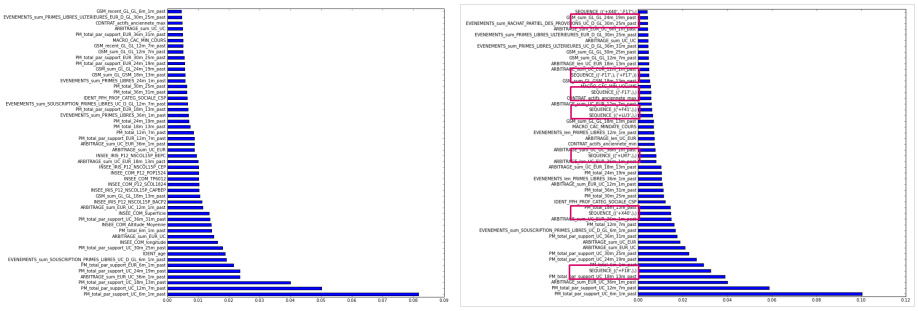


Figure 11: Representation of the 50 more "important" features given by the random forest algorithm. The importance is measured by counting around the occurrence these features appear on top of the random threes. The result without introducing sequence mining features is displayed on the left side and the one with the sequence mining features appears on the right side.

5.4 The performance of the model

Figure 12 shows the global performance of the model with respect to ROC AUC and LIFT metric run for the lapse target. The model run here is the meta model obtained from a linear regression of outputs predictions of ensemble methods in the first stage.

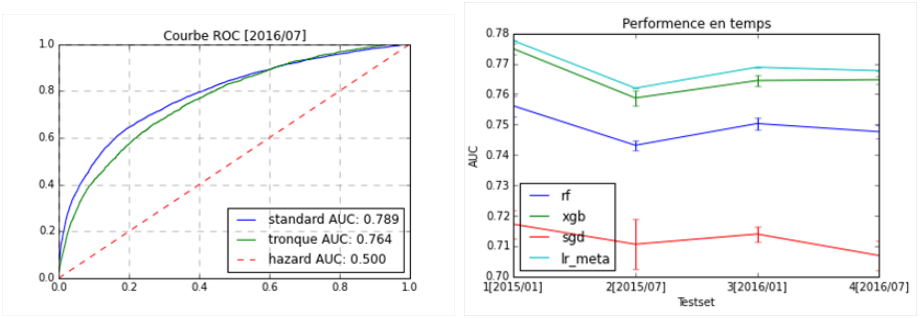


Figure 12: ROC AUC measured in different time period is displayed on the right and run on the $[2016_{07}, 2016_{12}]$ time period on the left. We can notice the stability of the prediction performances in time

We compute the distribution of the mathematical reserve in figure 13, since the underlying features from mathematical reserve appears as the most important feature in the feature selection graph - *see figure 11*. We can see that the distribution from the population predicted by the model is biased toward the left (in comparison to the distribution of population that really performs an adverse action into the considered period). That means the model seems to capture frequent adverse actions on low mathematical reserves. To test this point, the same model is applied to the same portfolio but with a truncation on the "target population" : the label 1 is flagged only for adverse actions above 40 000 € - *see figure 14 where we can notice that the mean of the global distribution of the label goes from about 10% to 4% - namely 5 000 and 2 000 customers.* The light degradation of the ROC AUC (from 0.789 to 0.764 on the $[2016_{07}, 2016_{12}]$ time period) indicates that big lapses (above 40 000 €) are more difficult to predict.

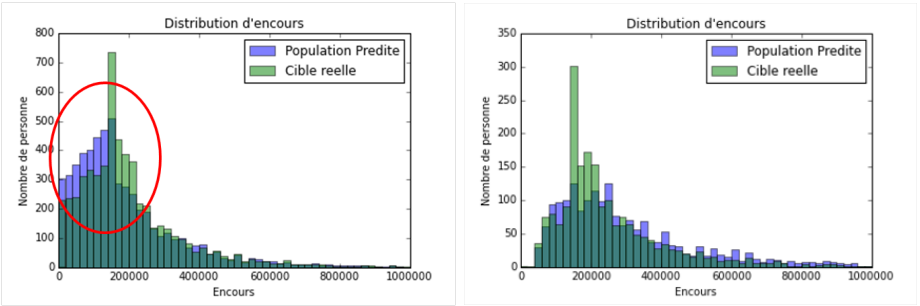


Figure 13: Distribution of customers with respect to the mathematical reserve

However, **we can conclude** from the perceptible stability of the ROC AUC with respect to time - *see the illustration displayed on the right from figure 12* - **that the predictive meta model gives stable results**.

In addition, we can see on figure 15 that the predictions given by the model provides relative strong results. With a selection of 10% of the population - *namely about 5 000 customers* - (with the best scores on targeting the label 1), **the model returns about 45%** - namely about 2 250 - **of right true positives**. This precision is of course higher with the proportion of prospects decreasing - **With a selection of 1% of the population, the precision is about 80%**. Besides, **the same representation run on the population with a truncated target shows that about 37%** (about 300 of the 800 targeted clients - *namely the "primo racheteurs"*) of the predicted population are customers that have never made an adverse action before. The precision

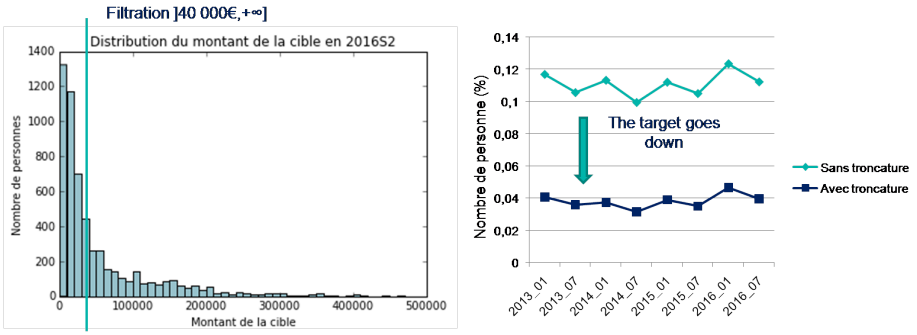


Figure 14: Representation of the distribution of amounts of surrenders recorded in the second semester of 2016 on the left side. The mean of the distribution target fall from 10% to 4% (on the right)

of the model remains high with a precision of 40% (about 800 of the 2 000 clients who really perform the adverse action).

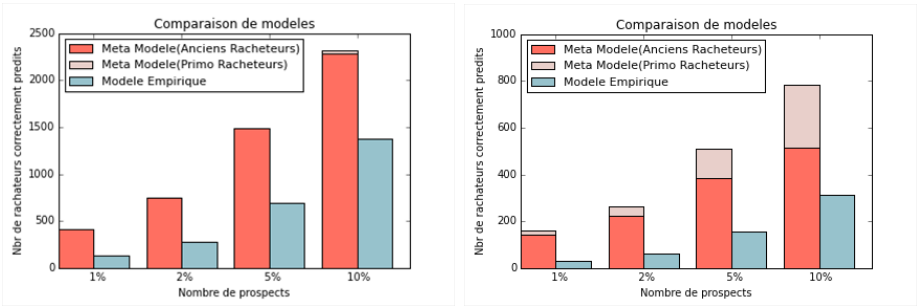


Figure 15: Representation of the number of right predictions from a selection between 0 and 10% of the population. The graph on the left illustrate the performance of the predictive model in comparison to the empiric model. The graph on the right represents the same illustration run on the population with a truncated target

Last, the recall - *the proportion of the real adverse actions is recalled by the model, finely illustrated from wikipedia on figure 16* - **remains high at 40% with respect to a 10% selection of the whole population of the portfolio**.

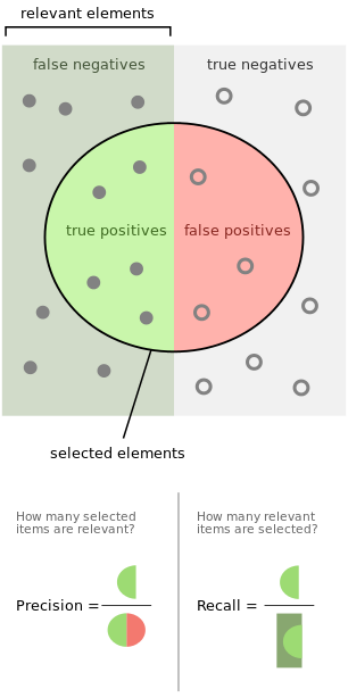


Figure 16: From wikipedia, illustration for the *precision* and *recall* definition

In conclusion, since actuaries are familiar on handling quantitative tools, we gradually show the power of machine learning approaches in terms of value added : improving the knowledge on the managed portfolio. We hope that actuarial community will find from this paper a good way to explore and adapt these methods.

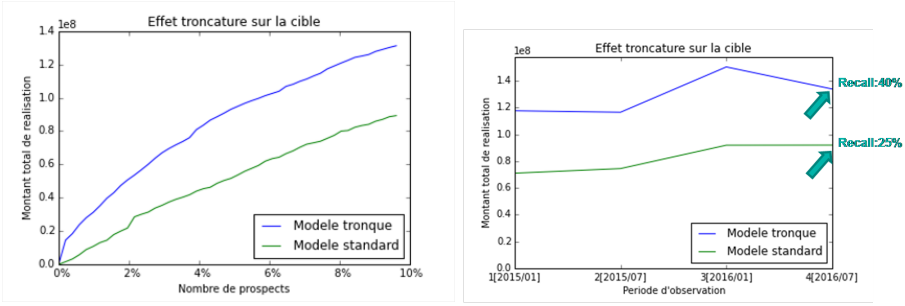


Figure 17: Representation of the amount stemming from the positive predictions linked to lapses on the left side. The prediction based on the truncated target allows focusing on more amounts in risk vision. The amount of adverse actions correctly predicted remain stable in times and the recall stay relatively high level at 40%

A Random Forest Algorithm

The following description comes from [FRI01a]

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $Z*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C_{rf}^B}(x) = majorityvote\{\hat{C}_b(x)\}_1^B$.

B Gradient Boosting Algorithm

The following description comes from [FRI01a]

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(y_i, \gamma)$
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute
   $$r_{im} = -\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}|_{f=f_{m-1}}$$
   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$,
   (c) For $j = 1, 2, ;J_m$ compute
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$
   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
3. Output $\hat{f}(x) = f_M(x)$

C Understanding of the machine learning problem

Let us consider two classical classes of algorithms broadly used - the *Linear Regression and the Decision Tree*. We put them in parallel and bring their specifications back to the same mathematical formulation. When this wording is easy to find for Linear Regression, it seems to be not the case for Decision Tree algorithm for which the following formulation is taken from [TH08] p305.

- **Example 1** : The Linear Regression

  (1) If we consider a linear relation for f, then we have the Linear Regression problem and we can write the linear connection between x and θ:

  $$f(x) =< x; \theta >= \sum_{i=1}^{d} x_i.\theta_i$$

  For some parameter θ to be assessed from experienced data $\{x_i, y_i\}$. We can notice an important remark : **the linearity is here not necessarily with respect to the original features**. It's possible to choose x_i beyond the original features observed - *feature engineering*. To illustrate this point, let's consider a set of 2 features $X = \{x_1, x_2\}$ and the corresponding experienced label y_0. Resolve this problem with Linear Regression doesn't mean without any nonlinearity effect between y_0 and X with respect to f function. In fact, by adding $\{x_1^2, x_2^2, 2.x_1.x_2\}$ features to our original database $X = \{x_1, x_2\}$, one can fit f to be linear on $\{x_1, x_2, x_1^2, x_2^2, 2.x_1.x_2\}$ but, as a result, nonlinear on $\{x_1, x_2\}$ features. We can already sense the importance of feature engineering.

  (2) The loss function could be chosen among some classical one. The most frequent are the L_p **loss** function $l(y, f(x)) = |y, f(x)|^p$ with $p \geq 1$ suitable for continuous labels and the **logistic loss** function (Figure 18) that is appropriate for binary classifications :

  $$l(y, f(x))_{LF} = -log[\sigma(f_\theta(x))] \; if \; y = 1 \; and \; -log[1 - \sigma(f_\theta(x))] \; if \; y = 0$$

  As we are interested in the classification problem, let's recall that the use of the logistic function $\sigma(.)$ allows to convert the output function $f_\theta(x) =< x, \theta >$ into $[0, 1]$ interval.

  $$\sigma(z_\theta) = \frac{1}{1 + \exp(-z_\theta)} \; , \; With \; some \; probability \; function \; characteristics$$

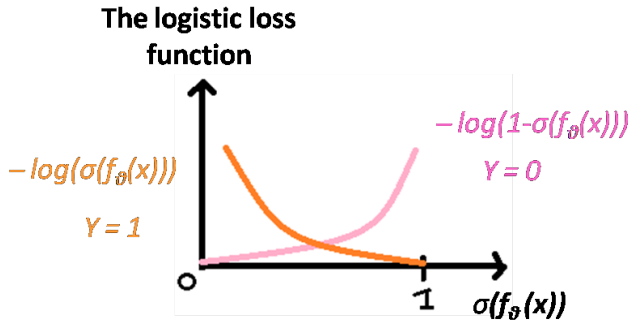


Figure 18: The logistic loss function $l(y, f(x))_{LF}$ for any $\sigma(f_\theta(x))$ in [0,1]

The definition of the loss function on Figure 18 has a few interesting and desirable proprieties : First when $y = 1$ and $f_\theta(x)$ is estimated to be 1, we read on the "orange" graphic that the cost function is 0. Second, when $y = 1$ and $f_\theta(x)$ is estimated to be 0, then the "orange" graphic shows that the cost function tends to be infinitely high. The same observation can be made on the pink curve when $Y = 0$.

The **statistical interpretation** of using the σ_θ function - *we mention earlier above that the machine learning reasoning is not assuming any condition.* - is as following : By assuming that the occurrence probability of an adverse management act Y follows a Bernoulli motion -

$$P(Y = y|X = x) = Bernoulli(\sigma_\theta(x)) - Then\ we\ get$$
$$P(Y = y|X = x) = \sigma_\theta(x)^y.(1 - \sigma_\theta(x))^{1-y}.$$

The classical minimization of the log-likehood wrt to θ parameter gives -

$$\frac{\delta(\sum_{i=1}^{d} logP[(Y = y_i|X = x_i)])}{\delta\theta} = 0$$

and stemming for that, we get an estimation of the individual probability $\hat{p}_j^\theta$ to have label $y_j = 1$ from x_j observation. We have just to apply the logistic function $\hat{p}_j^\theta = \frac{1}{1+\exp(-<x_j;\hat{\theta}>)}$. In addition, $\hat{\theta}_i$ coefficient corresponding to the i^{th} feature $x_{j_{(i)}}$ provides information about the contribution of that feature on the probability assessment - *a large positive $\hat{\theta}_i$ means a positive correlation wrt to the label and a negative one means the opposite scenario.*

A simple "Maximum A Priori" rule gives the prediction of the label 1 if $\hat{p}_j > Threshold$ (Let's say Threshold $= 1/2$) and 0 otherwise. This comfort on describing and analyzing results implies the large use of the Logistic Regression classifier.

(3) As the optimization issue - *hugely detailed in literatures* - is not the point we are focused on in this paper, we do not develop this point so far. The problem is addressed as follows :

$$\hat{\theta} \in argmin_{\theta \in R^d} \quad R_n^\theta = [\frac{1}{n}\sum_{i=1}^{n} l(y_i, < x_i, \theta >))]$$

Let's just keep in mind that most of the optimization algorithms need a smoothness and convexity assumption on R_n^θ function on order to avoid local minimums. Many ameliorations of the classical batch gradient descent algorithm do exist, see [Bot04]. The principal is to The stochastic gradient descent will be used later as the reference in assessing the performance of predictions.

- **Example 2** : The Decision Tree

(1) Decision Trees recursively partitions the space χ such that the samples with the same labels are grouped together. It rests on fitting a very simple function - *like a constant* - in each sample and by the way assumes that f(.) is well approximated by a locally piecewise-constant function - *with different values of the constant on each region of the space χ.* The variable - *on which the split will be applied* and the split-point are chosen to achieve the best fit - *according to some conditions.* A Three with L leaves will be a set of L rules R_k. For each leaf k, a rule R_k is associated to a subset $\chi_k \subseteq \chi$. Then we write :

$$\hat{f}(x) = \sum_{k=1}^{L} \hat{\theta_k}.R_k(x)$$

$R_k = I[x \in \chi_k]$ is worth 1 if x belongs to the space is worth 1 if x belongs to the space χ_k and zero if not.

If we adopt as criterion minimization a given loss function l, then we write $\hat{\theta_k} = l(y_i, f(x_i))$. It's easy to see that $\hat{\theta_k} = E_n[y_i/x_i \in \chi_k]$ is the empirical mean of Y on the node k .
Then the main concern is to choose the variable split and the split-point. Starting with all of the data, consider a splitting variable j and split point s, and define the pair of half-planes

$$R_1(j, s) = \{X/X_i \leq s\} \text{ and } R_2(j, s) = \{X/X_i > s\}$$

We seek the splitting variable j and split point s that solve

$$min_{(j,s)}[min_{\theta_1}[\sum_{x_i \in R_1(j,s)} l(y_i, \theta_1)] + min_{\theta_2}[\sum_{x_i \in R_2(j,s)} l(y_i, \theta_2)]]$$

For any choice j and s, the inner minimization is solved by

$\hat{\theta_1} = E_n[y_i/x_i \in R_1(j,s)]$ and $\hat{\theta_2} = E_n[y_i/x_i \in R_2(j,s)]$

Having found the best split, the data are partitioned into the two resulting regions and the splitting process is repeated on each of the regions.

Following the above recall, **we can retain that supervised learning problems stands for looking at class of estimators and loss functions** adapted to the experienced data. We can also notice that the solutions of these problems go through optimization concerns. The good news for practitioners is many of these algorithms have already been implemented, regularly improved and totally free on *@R or PYTHON* language packages. The main leverages on taking full advantages from these works are (1) to write out onto machine learning problem, (2) to transform in a suitable way the input database before the learning step and (3) eventually to improve performance of predictions.

The first point is not so trivial. With the previous notation, we need to attribute and output label y_i to each set of features $\{x_i\}$ in order to have available the training database $\{x_i \ ; \ y_i\}$. This **expert information telling the model what is "1" and what is "0" regarding to business point of view is not always available and have to be built**.

D The Sequence Mining

D.1 Problem Definition

Let $I = \{i_1, i_2, ...i_m\}$ be a set of literals called items - *it may be for instance a single article bought by a customer if we substitute ourself to a large distribution store when mining daily transactions on their stores.* An item set X - *also called transaction and can be understood as receipt if we pursue with the retail industry analogy* - is a set of items ($X \supseteq I$). A sequence $s = \langle t_1, t_2, ..., t_n \rangle$ is an ordered set of item sets. The length of s is defined as the number of items contained in s. We use $|s|$ to represent the length of s. A sequence of length k is called a k-sequence.

Definition 1 *Consider two sequences* $s_1 = \langle a_1, a_2, ..., a_n \rangle$ *and* $s_1 = \langle b_1, b_2, ..., b_n \rangle$. s_2 *is a subsequence of* s_1 *if there exist integers* $j_1, j_2, ..., j_l$, *such that* $1 \leq j_1 < j_2 < ... < j_l \leq n$ *and* $b_1 \subseteq a_{j_1}, b_2 \subseteq a_{j_2}, ..., b_n \subseteq a_{j_n}$. *We represent this relationship by* $s_2 \sqsubseteq s_1$.

A database D consists in a collection of sequences. The support of the sequence s is defined as the fraction of all sequences in D that contain s. We use $supp(s)$ to denote the support of s. If the support of s is no less than a user specified support threshold, s is a frequent sequence.

Theorem 1 $\forall$ *two sequences* s_1, s_2, *if* $s_2 \sqsubseteq s_1, supp(s_1) \leq supp(s_2)$

In our application, the objective is (1) to identify the frequent subsequences among all possibilities of combinations (the join phase and the prune phase) and (2) to count them in a suitable way avoiding a heavy computational time (the count phase).

D.2 Frequent Subsequence Mining Algorithm

The Generalized Sequential Patterns (GSP) (see [SRI96]), is a classical frequent subsequence mining algorithm. GSP is based in a "Apriori" algorithm. It applies Apriori's join phase and prune phase for subsequence candidates and mix it with a clever candidate count phase. It also incorporates some generalization features of sequence mining that are not experimented in this project, such as taxonomy, time constraints and sliding windows as explained in [SRI96]. GSP is proved to be 30% to 5 times faster than Apriori because its candidate count phase applies a hash tree search and does not require a prior dataset transformation.

We will present in the next section the join phase and prune phase. Notice that GSP share the same frame here with an optimization program on the count phase.

D.2.1 Apriori

Apriori algorithm is originally conceived for frequent item set mining. It uses a "bottom up" approach, where frequent subsequence are extended one item at a time (this step is known as candidate generation), and groups of candidates are tested against the data.

Algorithm 1 Apriori($dictionary, dataset, minSupport$)

CandidateSet C_1 := All single items in the *dictionary*
CountPhase : Scan *dataset* to get support of every sequence in C_1
FrequentSetF_1 := every sequence in C_1 whose support > *minSupport*
i := 1
while $F_i \neq \emptyset$ **do**
  CandidateSet C_{i+1} := **JoinPhase**(F_i)
  CandidateSet C_{i+1} := **PrunePhase**(F_{i+1},$F_1 \cup F_2 \cup \ldots \cup F_i$)
  CountPhase : Scan *dataset* to get support of every sequence in C_{i+1}
  FrequentSet F_{i+1} := every sequence in C_{i+1} whose support > *minSupport*
  i := i+1
end while
return $F_1 \cup F_2 \cup \ldots \cup F_{i-1}$

D.2.2 Join Phase

Join phase relays on the following definition and theorems that can be found in [SRI96].

Definition 2 *Given a sequence $s = \langle s_1 s_2 \dots s_n \rangle$ a subsequence c, c is a contiguous subsequence of s if any of the following conditions hold:*

1. *c is derived from s by dropping an item from either s_1 or s_n.*
2. *c is derived from s by dropping an item from an element s_i which has at least 2 items.*
3. *c is a contiguous subsequence of c', and c' is a contiguous subsequence of s.*

Theorem 2 *Any sequence s that contains a sequence c will also contain any contiguous subsequence of c.*

We generate candidate sequences of C_{i+1} by joining F_i with F_i. A sequence s_1 joins with s_2 if the subsequence obtained by dropping the first item of s_1 is the same as the subsequence obtained by dropping the last item of s_2. The candidate sequence generated by joining s_1 with s_2 is the sequence s_1 extended with the last item in s_2. The added item becomes a separate element if it was a separate element in s_2, and part of the last element of s_1 otherwise. When joining F_1 with F_1, we need to add the item in s_2 both as part of an item set and as a separate element, since both $\langle \{x, y\} \rangle$ and $\langle \{x\})(\{y\}) \rangle$ give the same sequence $\langle \{y\} \rangle$ upon deleting the first item. (Observe that s_1 and s_2 are contiguous subsequences of the new candidate sequence.)

D.2.3 Prune Phase

Prune phase relays on a fundamental property of frequent subsequence:

Theorem 3 *If a subsequence s is frequent, then all subsequence of s must be frequent.*

Theorem 3 shows that if a subsequence is to be found in F_{i+1}, all its subsequence can be found in $F_1 \cup F_2 \cup ... \cup F_i$. Thus, in the prune phase, we check again that all subsequences from candidates generated from the join phase can be found in former frequent set, otherwise we prune the candidate.

D.3 Sequential rule generation

Frequent subsequence mining algorithm can help us find frequent sequential patterns in our database. However, we know that a sequence appearing frequently in a database is not sufficient for making prediction. For example, it is possible that an event c appears frequently after some events a and b but that there are also many cases where a and b are not followed by c. In this case, predicting that c will occur after a, b according to a frequent subsequence $\langle a, b, c \rangle$ could be a huge mistake. Thus, to make predictions it is desirable to have patterns that indicate how many times c appears after a, b and how many times it does not. To address this problem, sequential rule is often used in data mining for prediction purpose. Since it's easy to understand and visualize, it's also popular as a presentation tool in business context.

Yet, several mathematical formulations for sequential rule are found in published papers. A similar definition to our problem can be found in [FV12], along with a comparison to the association rule mentioned in [RA95]. Nevertheless, their formulation for sequential rule do not matches exactly our problem. The left hand side of [FV12] is by default transformed to a non-ordered set of items so as to capture rules between items. This procedure ignore altogether the sequential order of the left hand side's transaction and the possible information about the number of times a same item occurs.

In the next section, we will formulate our proper definition of sequential rule that is more adaptive to our problem - *Mining management acts on our savings portfolio* - , followed by an algorithm to calculate it through mined frequent subsequences.

D.3.1 Sequential rule definition

Under the definition of sequence in Definition 1, a sequential rule is a relationship between two sequence s_1 and s_2, noted as $s_1 \Rightarrow s_2$. The interpretation of a rule $s_1 \Rightarrow s_2$ is that if s_1 occurs in a sequence, s_2 will occur afterward from the same sequence.

(1) For a given dataset D. Three measures are constructed for each rule $s_1 \Rightarrow s_2$. The support of the rule $supp(s_1 \Rightarrow s_2) = \frac{supp(\langle s_1, s_2 \rangle)}{|D|}$, where $\langle s_1, s_2 \rangle = \langle t_1, t_2, \ldots, t_n, m_1, m_2, \ldots, m_l \rangle$ for $s_1 = \langle t_1, t_2, \ldots, t_n \rangle, s_2 = \langle m_1, m_2, \ldots, m_l \rangle$. The support can be interpret as the frequency of occurrence of this rule.

(2) The confidence of a rule $conf(s_1 \Rightarrow s_2) = \frac{supp(\langle s_1, s_2 \rangle)}{supp(s_1)}$ indicates its probability of realization. Notice that $s_1 \sqsubseteq \langle s_1, s_2 \rangle$, with Theorem 1 we can deduce that $supp(s_1) \geq supp(\langle s_1, s_2 \rangle)$. The rule's confidence varies form 0 to 1, therefore it's a probability.

(3) The lift of a rule $lift(s_1 \Rightarrow s_2) = \frac{conf(s_1 \Rightarrow s_2)}{supp(s_2)}$ indicates the prediction power of the rule with respect to a random model. For example, For example, suppose a sequence has an average occurrence rate of 5%, but our rule can deduce a occurrence rate of 20% after having observed a certain history. Then this rule would have a lift of 4.0 (20%/5%). In a way, lift indicates the significance of a rule : the higher lift is, the more interesting the rule is. On the other hand, a rule with lift around 1 doesn't help much to predict.

Theorem 4 *For any sequential rule $r : s_1 \Rightarrow s_2$ in a database D, the relation $conf(r) \geq supp(r)$ holds.*

These three measures can help us prune any uninteresting rule: if the support is too low, the rule is not general enough; if the confidence is too low, the rule is not powerful enough and if the lift is too low, the rule does not help much our prediction. For this reason, thresholds can be fixed to remove these rules $(minSupport, minConfidence, minLift)$. Notice that we always have confidence greater than support, so it's meaningless to set $minConfidence \leq minSupport$.

D.3.2 Sequential rule generation algorithm

The algorithm for rule generation is conceived under the premise that frequent subsequences are already mined and output along with their supports all of which surpass the predefined threshold $minSupport = \rho$ (Algorithm 1) and that sequential rules to be generated have their supports greater than ρ. In this case, we can exploit our result from the frequent subsequences to generate easily sequential rules.

Algorithm 2 GenerateSequentialRules($FreqSeqList, SupportDict, minSupp, minConf, minLift$)

```
Sort FreqSeqList by lexicographic order
Rules = ∅
for S1 in Sorted FreqSeqList do
  L := sublist of Sorted FreqSeqList contains all elements after S1
  H1 := pop first element of L
  while H1 begins with S1 do
    S2 := remove S1 from the beginning of H1
    R1 = ComputeMeasures(SupportDict,S1,S2,H1)
    if R1 satisfies minSupp, minConf, minLift then
      Put (S1, S2, R1) in Rules
    end if
    H1 := pop first element of L
  end while
end for
return Rules
```

The algorithm uses frequent subsequences and their support to generate association rules. The time complexity is $O(N^2)$ in the worst case. In practice, the complexity is almost linear. Key

element of this algorithm is lexicographic sort of the frequent subsequence list. Definitions and the proof of the completeness of this algorithm are presented.

Definition 3 *A is a finite set of transaction, totally ordered. Given two different sequences of the same length* $s_1 = \langle a_1, a_2, \ldots, a_k \rangle$ *and* $s_2 = \langle b_1, b_2, \ldots, b_k \rangle$ *where* $\forall i \leq k$, $a_i, b_i \in A$, *the* s_1 *is smaller than* s_2 *for the lexicographical order* ($s_1 <_{lexo} s_2$), *if* $a_i <_A b_i$ *(for the order of A), for the first i where* a_i *and* b_i *differ. If two sequences are not the same length, the shorter sequence is padded at the end with enough "blanks" (a special symbol that is treated as smaller than every element of A).*

For any frequent sequential rule $s_1 \Rightarrow s_2$ ($supp(s_1 \Rightarrow s_2) \geq \rho$), all following statements have to be true to prove the completeness of this algorithm.

1. s_1,s_2,$\langle s_1, s_2 \rangle$ are all frequent subsequence
2. $\langle s_1, s_2 \rangle >_{lexo} s_1$
3. $\forall s'$ tel que $\langle s_1, s_2 \rangle >_{lexo} s' >_{lexo} s_1$ beings with s_1, i.e. $\exists s''$ tel que $s' = \langle s_1, s'' \rangle$

Proof 1

1. *By definition of the rule support,* $supp(\langle s_1, s_2 \rangle) \geq \rho$, *so clearly* $s_1 \sqsubseteq \langle s_1, s_2 \rangle$. *By Theorem 1,* $supp(s_1) \geq supp(\langle s_1, s_2 \rangle) \geq \rho$
2. *By Definition 3,* s_1 *needs to be padded with blanks at the end to match the length of* $\langle s_1, s_2 \rangle$. *After the padding, we see the first transaction that differ between the two, is blank and the first transaction of* s_2. *By definition blank is smaller. So* $\langle s_1, s_2 \rangle >_{lexo} s_1$.
3. $\langle s_1, s_2 \rangle >_{lexo} s' >_{lexo} s_1$. *Suppose the number of transactions in* s_1 *is* l_{s_1}. *After padding each sequence, by Definition 3, the first* l_{s_1} *transactions of* s' *(noted* s_{commun}*)* $\geq_{lexo}$ *the first* l_{s_1} *transactions of* s_1, *and also* $s_{commun} \leq_{lexo}$ *the first* l_{s_1} *transactions of* $\langle s_1, s_2 \rangle$. *Thus* $s_1 \leq_{lexo} s_{commun} \leq_{lexo} s_1$, $s_{commun} = s_1$. *Therefore, define* s'' *as the rest of* s' *after* l_{s_1}, *we have* $s' = \langle s_1, s'' \rangle$

By Proof 1, we prove that for each frequent rule, the left hand side of the rule will be found in the frequent subsequence list and the while loop will not stop until every potential rule is mined. So the algorithm is complete.

D.3.3 Sequential rule for prediction

Sequential rule can be used to build prediction model. [YAN04] used directly mined sequential rule to predict the next move of web users. They proposed several solutions and criterion for rule selection and compared respectively their performance. However, these methods are more adapted to their project where data is scarce apart from the user's browsing history. In this project, other variables are also important for prediction, such as age, total savings in the contract that are not a part of transaction data. To solve this problem, [SRI96] proposed ideas to mine the personal attributes. They divided each attributes, categorical or quantitative, into items so as to transform attributes into item sets and then they mined association rules out of these item sets. The same procedure can be applied to sequential rule, too. Their idea is later developed by [GYE01] who used a fuzzy method instead of simple divisions to mine quantitative attributes.

However, all these rules are not well adapted because the more the variables are abundant, the more complicated the model will be. The capacity of generalization is also weak as the rule may change overtime, and two much parameters needs to be predefined. In the next section, we propose another **scheme to incorporate sequences into prediction model**.

D.4 Feature generation

Transform frequent subsequences into features is a very convenient way to build sequence learning for predictive models. It's more flexible as other non-temporal data can all be combined as features into the prediction and all conventional classification methods would be applicable. Several methods are synthesized by [YAN04] to transform sequence into features.

1. k-gram based feature selection : k-gram is a short sequence segment of k consecutive symbols. Given a set of k-grams, a sequence can be represented as a vector of the presence and the absence of the k-grams or as a vector of the frequencies of the k-grams.

2. pattern-based feature selection : [KUD98] proposed a pattern-based feature selection method. The features are short sequence segments which satisfy the following criteria (1) frequent in at least one class (2) distinctive in at least one class and (3) not redundant. Criterion (2) means a feature should be significantly correlated with at least one class. The redundancy in Criterion (3) can be defined in the way of feature specification and feature generalization.

3. local and global feature selection : Although subsequences are informative features, they can only describe the local properties of a long sequence. [AGG02] modify wavelet decomposition to describe a symbolic sequence on multiple resolutions. With different decomposition coefficients, the wavelet represents the trends in different range of intervals, from global to local.

The taxonomy listed here contains numerous methods in term of variety, and they are all applicable to symbolic sequences, which is our case. Several questions are helpful to decide between these methods.

- Should feature selection be integrated within the process of constructing the classifier or a separate preprocessing step?
- In which scope does feature selection reflect the sequential nature of a sequence, local or global?
- Which criteria should be used for selecting features, such are distinctiveness, frequency, and length?
- Should matchings be exact or inexact with gaps?

The method that conducts on results presented on Section 5 of this paper uses **frequent subsequences to generate features**. To answer the questions above, the method is a preprocessing step at a local scope, using frequency as criterion with exact matching.

E Cross blending for a better stacking performances

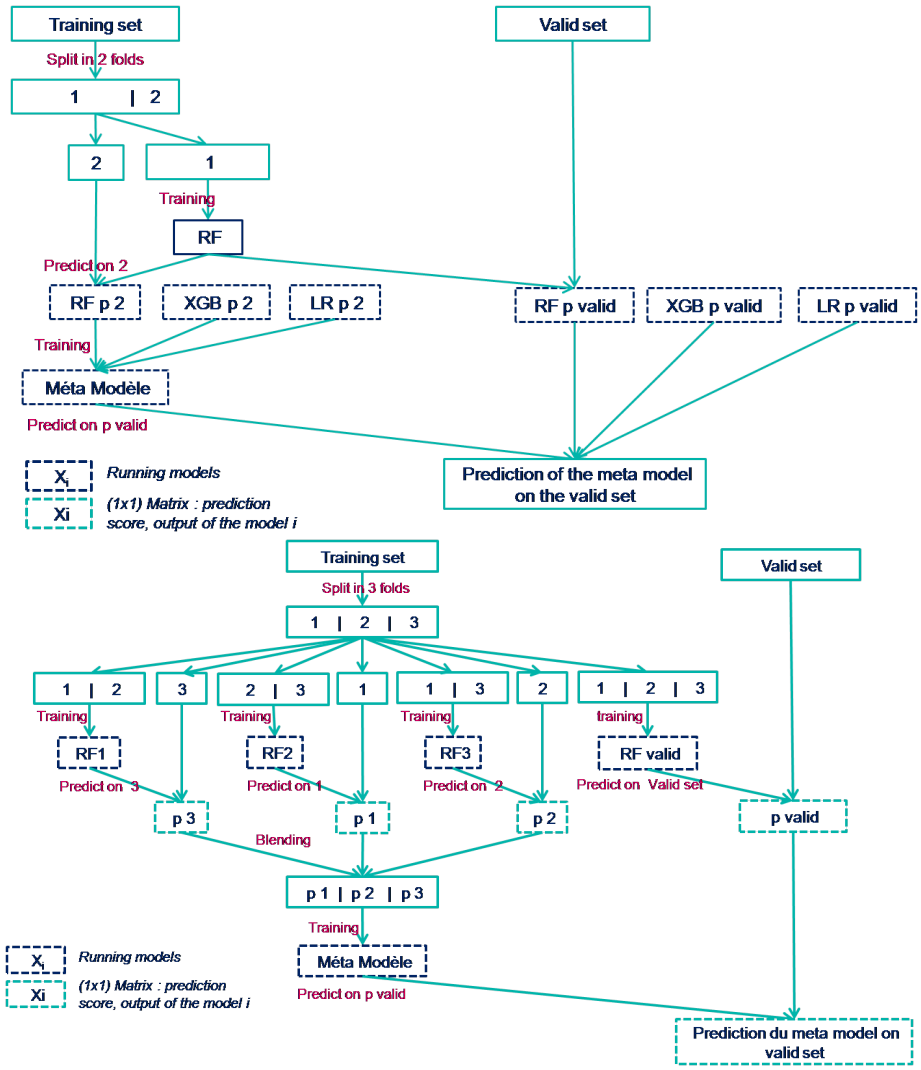


Figure 19: The classic way of blending predictive models - *upper side* - is not so efficient since it conducts to a dilemma. At the data split step, we have to choose between : keep more data for data split 1, implying a poor meta model performance OR Keep more data for data split 2, implying a poor performance for the first stage models. The new approach - *down side* - of blending the meta-model allows us to preserve the complete dataset 1/2/3 for training the meta model, while ensuring a good performance for basic models

References

[ACP15] ACPR. Analyse de l'exercice 2015 de préparation à solvabilité ii. (56):25, 2015.

[AGG02] Charu C. AGGARWAL. On effective classification of strings with wavelets. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.*, pages 163–172, 2002.

[Bot04] Leon Bottou. Stochastic learning. *Springer*, pages 146–168, 2004.

[CF15] Osmar Castrillo-Fernandez. Web scraping: Applications and tools. *European Public Sector Information Platform*, 2015, 2015.

[DC03] HOLTE Robert C. et al. DRUMMOND Chris. C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. *Workshop on learning from imbalanced datasets*, II, 2003.

[FRI01a] HASTIE Trevor et TIBSHIRANI Robert FRIEDMAN, Jerome. The elements of statistical learning. *Springer, Berlin : Springer series in statistics*, 2001.

[FRI01b] Jerome H. FRIEDMAN. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[FV12] FAGHIHI Usef NKAMBOU Roger et al FOURNIER-VIGER, Philippe. Cmrules: Mining sequential rules common to several sequences. *Knowledge-Based Systems*, 25(1):63–76, 2012.

[GYE01] Attila. GYENESEI. A fuzzy approach for mining quantitative association rules. *Acta Cybern*, 15(2):305–320, 2001.

[KUD98] Haym KUDENKO, Daniel et HIRSH. Feature generation for sequence categorization. *AAAI/IAAI*, pages 733–738, 1998.

[LIU05] Ya-Yueh LIU, Duen-Ren et SHIH. Hybrid approaches to product recommendation based on customer lifetime value and purchase preferences. *Journal of Systems and Software*, 77(2):181–191, 2005.

[RA95] Ramakrishnan Srikant Rakesh Agrawal. Mining sequential patterns. *IEEE Computer Society Washington*, pages 3–14, 1995.

[SD04] BERNARD ZENKO SASO DZEROSKI. Is combining classifiers with stacking better than selecting the best one. *Springer, Kluwer Academic Publishers.*, Machine Learning(54):255–273, 2004.

[sld16] scikit-learn developers. Supervised learning. *http://http://scikit-learn.org*, Online Source, 2016.

[SRI96] Rakesh SRIKANT, Ramakrishnan et AGRAWAL. Mining quantitative association rules in large relational tables. *Acm Sigmod Record*, pages 1–12, 1996.

[TH08] Jerome Friedman Trevor Hastie, Robert Tibshirani. The elements of statistical learning. *Springer*, 2008.

[TS15] Marc Rehmsmeier Takaya Saito. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0118432#pone-0118432-g005*, Online Source, 2015.

[YAN04] LI Tianyi et WANG-Ke YANG, Qiang. Building association-rule based sequential classifiers for web-document prediction. *Data mining and knowledge discovery*, 8(3):253–273, 2004.