Machine Learning : quelles opportunités de pilotage pour le passif d'un assureur vie ?

B. TESSIAUT[1], **N. DUSSERRE**[2,+]

[1] *Actuaire IA chez Command Strategy Advisory*
[2] *Ingénieur en informatique chez Command Strategy Advisory*
[+]ont aussi contribué : D. AMICHIA, T. BELLINI, M. DEMERGERS et A. GIRAULT.

ABSTRACT

Dans un contexte économique et réglementaire en perpétuelle mouvance, le pilotage ALM de l'assureur vie doit nécessairement faire intervenir une optimisation de la structure de son passif.

A cette fin, cette contribution propose d'expérimenter et de comparer trois méthodes de *Machine Learning* : *Random Forest*, *Gradient Boosting* et *Support Vector Machine*.

Dans un premier temps, chaque méthode d'apprentissage sera calibrée sur une base représentative d'un portefeuille d'assurance vie (âge, salaire, profession, encours, ...), et comportant pour chaque client la notation associée au risque de son support (indicateur synthétique de risque (ISR) de la norme européenne PRIIP's).

Le protocole de calibrage sera explicitement détaillé pour chaque algorithme, afin d'étudier les opportunités d'amélioration de la précision de ces modèles dans le cas d'une base de données multi-classes non équilibrée. Le *tuning* des hyperparamètres avec *grid search*, la phase de *features selection* ainsi que les méthodes de *subsampling* seront ainsi introduites.

Dans un second temps, les performances des algorithmes post-calibrage seront testées, afin de valider leurs précisions quant à la détection de couples (profil de risque ; investissement) statistiquement incohérents, et ainsi proposer à ces clients une allocation en ligne avec les politiques ALM et commerciale de l'assureur.

Mots-clés : Gestion Actif-Passif, Restructuration du Passif, Profil de risque, Calibrage, *subsampling*, *Machine Learning*, *Random Forest*, *Gradient Boosting*, *Support Vector Machine*.

I. Introduction

La continuelle baisse des taux d'emprunt observée depuis plusieurs années sur les marchés européens fait émerger de nouveaux risques pour les assureurs vie français, déjà affaiblis par des taux garantis élevés dans un environnement toujours plus compétitif.

Cette dissonance entre un actif au rendement diminué par des taux bas (voire négatifs) et un passif aux taux garantis importants nécessite de nouvelles politiques de pilotage ALM. Des solutions à court terme peuvent être adoptées à l'actif (par exemple en se déportant sur des classes d'actifs plus risquées pour espérer un meilleur rendement) mais seule une optimisation en profondeur de la structure du passif peut avoir un impact concret sur l'équilibre du bilan de l'assureur.

Même s'il est possible de gérer les flux des nouveaux contrats émis en proposant des taux garantis en concordance avec la conjoncture économique de taux bas, il est nécessaire pour l'assureur de revoir l'allocation des contrats qui composent le stock de son portefeuille d'assurés.

Cette restructuration peut être effectuée en proposant à certains assurés d'orienter leur épargne vers une allocation favorisant les Unités de Compte si leur profil de risque est en cohérence avec cette suggestion (les UC étant plus rémunérateurs pour l'assureur, car sans garantie en capital). Nait alors la difficulté à discerner les assurés susceptibles d'avoir une allocation UC sous-pondérée par rapport à leur capacité à porter du risque.

L'émergence de nouvelles méthodes d'apprentissage statistique (aussi appelées *Machine Learning*) offre des outils performants de ciblage pour assister l'assureur dans cette tâche, à l'image des algorithmes de type *Random Forest*, *Support Vector Machine* et *Gradient Boosting*.

Ce papier propose un protocole de calibrage complet pour ces modèles afin de comparer leurs performances sur une base de données multi-classes non équilibrée, notamment dans le cadre de la restructuration du passif d'un assureur.

II. Méthodologie

L'objectif de cette publication n'est pas de réaliser une étude descriptive permettant de déterminer les contributions de chaque variable à un phénomène spécifique, mais plutôt de présenter une analyse complète d'un protocole de paramétrage de modèles de Machine Learning (ou ML) dans le cadre des enjeux évoqués en introduction.

II.1. Base d'apprentissage

La base de données étudiée est une base fictive, produite à partir de statistiques sur les épargnants de France métropolitaine. Ces données proviennent d'études réalisées sur les dix dernières années par l'Institut national de la statistique et des études économiques (Insee) et par la Fédération Française de l'Assurance (FFA).

La base de données est composée de 100 000 individus, selon la répartition suivante :

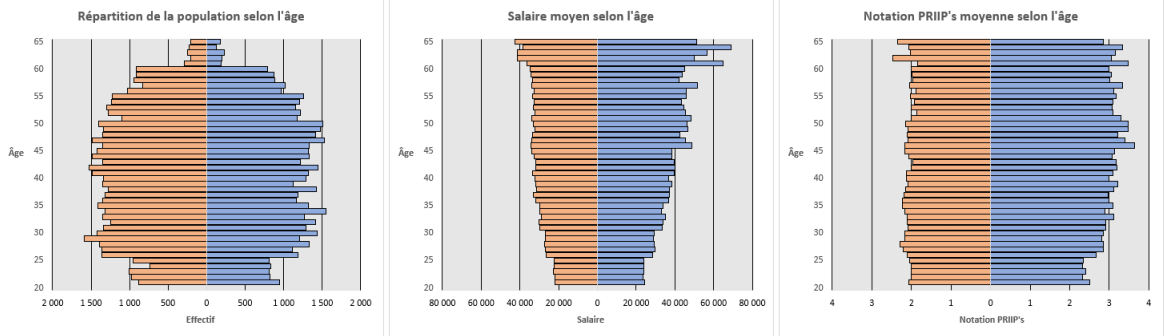

Figure 1. Répartition de la population d'assurés

Pour chaque individu, dix catégories de variables sont disponibles :

	Genre	Âge	CSP	Diplôme	Salaire
Type de variable	Qualitative	Quantitative	Qualitative	Qualitative	Quantitative
Modalités si dispo	H	De 21 à 65 (ans)	Ouvrier non qualifié	Etudes supérieures	
	F		Ouvrier qualifié	Niveau BAC	
			Employé	Niveau CAP, BEP	
			Profession intermédiaire	Aucun Diplôme	
			Cadres		

	Département	Statut matrimonial	Détention du contrat	Encours du contrat	Notation PRIIP's
Type de variable	Qualitative	Qualitative	Quantitative	Quantitative	Qualitative
Modalités si dispo	De 1 à 95	Célibataire	De 1 à 20 (années)		De 1 à 7
		En concubinage			
		Marié			

Figure 2. Descriptions des variables

La dixième variable, la notation PRIIP's, est la variable cible de l'étude. Depuis 2018, la norme PRIIP's (pour *Packaged Retail Investment and Insurance Products*) est la réglementation européenne qui régit les documents d'informations clés relatifs aux produits d'investissements (*Key information document*, ou KID). Ce document comporte notament une notation (appelée Indicateur Synthétique de Risque) qui représente la complexité et les risques de marché associés au produit. Cette notation s'étend de 1 à 7 : 1 pour un investissement avec pas ou peu de risque, 7 pour un risque important.

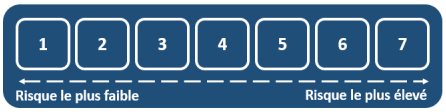

Figure 3. Échelle de l'Indicateur Synthétique de Risque selon la norme PRIIP's

Pour cette étude, la notation PRIIP's sera l'indicateur de risque du contrat d'assurance vie. L'hypothèse est faite que la rémunération de l'assureur augmente avec la notation PRIIP's, une notation plus élevée étant généralement associée à une part d'Unités de Compte plus importante dans l'allocation, ainsi qu'à des fonds avec une gestion active et des frais plus élevés.

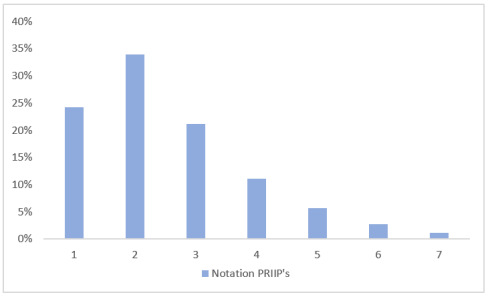

Figure 4. Répartition de la population selon la notation PRIIP's

II.2. Algorithmes de *Machine Learning*

Les modèles présentés ci-dessous peuvent être utilisés dans le cadre d'une régression ou bien d'une classification. La variable cible de cette étude étant une variable qualitative, une classification sera réalisée.

II.2.a. Random Forest (RF) :

Random Forest combine plusieurs arbres de décisions sous la forme d'un algorithme produisant de façon répétée des prédictions d'un même phénomène. Chacun de ces arbres de décisions est basé sur un échantillon de données et un nombre de variables sélectionnés aléatoirement.

II.2.b. Gradient Boosting (GB) :

Le modèle *Gradient Boosting* est aussi basé sur des arbres de décisions, mais diffère du modèle Random Forest du fait qu'il consiste à créer des arbres de décisions dans le but d'obtenir l'arbre le plus efficace. Le modèle va ainsi créer un premier arbre de décisions qu'il va ensuite corriger graduellement de manière séquentielle tout en pondérant les erreurs entre les différents facteurs. Le modèle final résulte de l'agrégation et l'optimisation des différentes branches.

II.2.c. Support Vector Machine (SVM) :

Les *Support Vector Machines* sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de classification et de régression. Ils ont pour but de séparer les données en classes à l'aide d'une frontière aussi « simple » que possible, de telle façon que la distance entre les différents groupes de données et la frontière qui les sépare soit maximale.

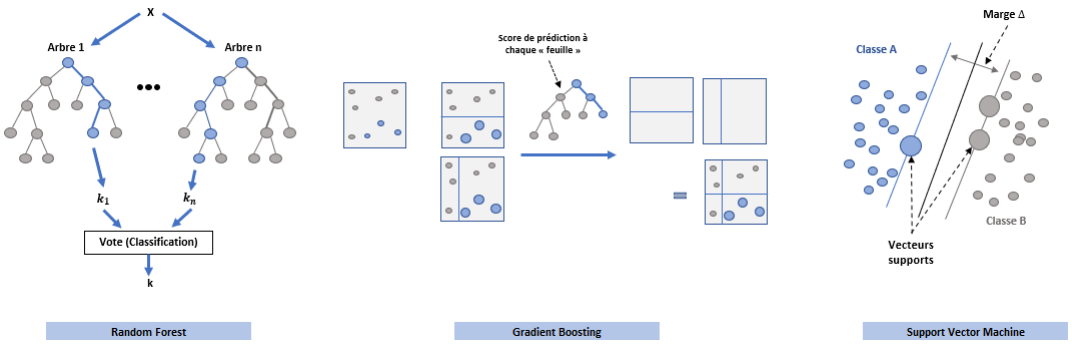

Figure 5. Description graphique des algorithmes de *Machine Learning*

II.3. Problématique spécifique

La problématique étudiée dans le cadre de ce papier est bien particulière. En effet, l'utilisation d'un algorithme de ML pour répartir une population en différentes catégories n'est pas nouvelle : les méthodes de classification supervisées sont courantes pour assister (voire remplacer) l'Homme dans des tâches faisant appel à des raisonnements analytiques.
Dans notre cas, l'objectif n'est pas de fournir une classification « avérée » des individus au sein de groupes de risques, parce que celle-ci n'existe pas. Il y a seulement des segmentations de la population selon différents critères (i.e. suivant différents paramétrages des algorithmes de ML), et il revient à l'analyste de déterminer comment les paramétrer pour répondre au mieux à sa problématique.

En d'autres termes, l'idée ici est de déterminer l'appétence au risque inhérente à chaque profil d'investisseur et tenter de discerner la tendance collective, afin de cibler les individus dont le choix isolé s'en éloigne, et réconcilier les choix individuels avec les choix collectifs.

II.4. Matrice de confusion, métriques de précision et optimisation des hyperparamètres

II.4.a. Matrice de confusion

La matrice de confusion est à la racine de toute analyse de performance d'un algorithme de ML. Elle présente la répartition de la population composant la base test selon les catégories proposées par l'algorithme : les colonnes indiquent la répartition des individus de la base testée selon leur label, et les lignes décrivent la répartition selon le label proposé par l'algorithme.

La diagonale de la matrice de confusion correspond aux individus classés dans la catégorie à laquelle ils appartiennent au sein de la base test.

		Base de test						
		1	2	3	4	5	6	7
Proposition de l'algorithme	1	4276	288	64	36	0	0	0
	2	472	6176	284	68	4	0	0
	3	36	204	3816	128	52	0	0
	4	0	0	116	1988	100	24	0
	5	0	0	16	28	1004	28	16
	6	0	0	4	4	16	504	8
	7	0	0	0	0	8	8	224

Logiciel : R / Modèle : RF / mtry : 8 / ntree : 1300

Figure 6. Exemple d'une matrice de confusion

L'analyse de la matrice de confusion est primordiale pour s'imprégner de la performance de l'algorithme et permet notamment de déterminer les métriques de précision.

II.4.b. Métriques de précision : accuracy, Kappa et balanced accuracy

Définition La précision (ou *accuracy*) correspond à la performance observée du modèle sur la base test, à travers la comparaison entre la classification proposée par l'algorithme et celle renseignée dans la base.

L'*accuracy* peut être observée sur la diagonale de la matrice de confusion : elle est calculée en divisant la somme des individus sur la diagonale par la totalité des individus composant la matrice.

$$\text{Accuracy} = [\sum_{i}^{nb.indiv.test} Si(Classif.Algo(i) = Classif.Base.test(i); 1; 0)] \times \frac{1}{nb.indiv.test}$$

Théoriquement, l'*accuracy* n'a de sens que si les données labellisées au sein de la base test ont été classées par un expert. Comme précisé précédemment, il n'existe pas de classification "idéale" pour chaque individu, et donc pas d'avis d'expert pour valider la cohérence de leurs investissements. Idéalement, la base test devrait être validée manuellement par plusieurs agents afin d'attester de la pertinence de la notation PRIIP's pour chaque profil. Néanmoins, c'est justement cette catégorie d'approche fastidieuse que les algorithmes de ML doivent permettre de contourner. Par conséquent, cette étude se concentre plutôt sur le fait de capter au mieux la tendance globale et de détecter les choix individuels qui s'en éloignent.

Définition Le Kappa (aussi appelé Kappa de Cohen) permet de discerner la composante de l'*accuracy* qui est à imputer au modèle, et non au hasard. En effet, un classement aléatoire de la base test peut éventuellement générer une bonne mesure d'*accuracy*, alors que celle-ci n'est pas associée à une bonne précision. Le Kappa propose de tenir compte de l'*expected accuracy*, qui correspond à la précision du modèle à imputer au hasard, par recoupement entre les répartitions de la base test et celle proposée par l'algorithme. Le Kappa correspond à la différence entre l'*accuracy* et l'*expected accuracy*, centrée sur la part des données dont la bonne classification n'est pas due au hasard :

$$Kappa = \frac{Accuracy - ExpectedAccuracy}{1 - ExpectedAccuracy}$$

Certes un Kappa élevé sera préféré, mais l'analyse de la répartition des classes proposée par le modèle, obtenue par l'intermédiaire de la matrice de confusion, sera tout aussi importante. En effet, un indicateur de précision de x% n'aura pas le même sens si toutes les classes comptent une précision de x% ou bien si cette performance est à imputer uniquement aux classes prépondérantes. D'où la nécessité d'utiliser une métrique robuste dans le cas de données non équilibrées, à savoir la *balanced accuracy*.

Définition La *balanced accuracy* correspond à la moyenne arithmétique de l'*accuracy* calculée pour chaque classe (i.e. un calcul de précision par colonne dans la matrice de confusion). La valeur ajoutée de cet indicateur réside dans le fait qu'il n'est pas biaisé par la prépondérance de certaines classes et permet ainsi d'analyser de façon robuste la précision d'un modèle dans le cas de données déséquilibrées.

II.4.c. Interprétation des métriques

Chaque résultat obtenu par l'application du protocole présenté dans cette contribution est propre à la base sur laquelle il est appliqué. Dans cette étude, l'attention doit être portée aux gains ou pertes sur les métriques associées aux différentes étapes du protocole de calibrage, et non à l'amplitude de ces mesures.

II.4.d. Hyperparamètres

Chacun des algorithmes étudiés va nécessiter le calibrage d'un certain nombre d'hyperparamètres. A la différence d'un paramètre classique (utilisé par le modèle pour réaliser une prédiction), un hyperparamètre va intervenir lors de la phase d'apprentissage et impacter le modèle sur sa complexité ou sa vitesse par exemple.

Les hyperparamètres associés aux différents modèles étudiés sont présentés dans le tableau ci-dessous :

	Hyperparamètres	Abréviation	Définitions
Random Forest	Number of trees	ntree	Nombre d'arbres générés par le modèle
	Number of features / node	mtry	Nombre de prédicteurs considérés à chaque création de nouvelle branche
Support Vector Machine	cost	C	Corrélé à la Variance du modèle : C élevé -> modèle plus précis mais moins généralisable
	gamma	γ	Corrélé au biais du modèle : γ élevé -> modèle moins précis mais plus généralisable
Gradient Boosting	eta	eta	Contrôle la quantité d'information utilisée dans chaque nouvel arbre
	colsample_by_level	col	Nombre de prédicteurs considérés à chaque création de nouvelle branche
	max_depth	max_d	Contrôle la profondeur maximale de l'arbre
	sub_sample	sub_s	Définit le type de Boosting utilisé
	gamma	γ	Amélioration minimale de l'erreur que chaque nouvelle branche doit respecter
	min_child_weight	min_child	Nombre minimal d'observations dans un nœud terminal
	nrounds	nro	Nombre d'itérations

Figure 7. Description des hyperparamètres pour chaque modèle

La recherche des hyperparamètres optimaux (ou phase de *tuning* en anglais), primordiale pour tout algorithme de ML, est la première étape du protocole présenté.

II.5. Protocole

Le protocole de calibrage présenté ci-après a été réalisé pour les algorithmes RF, GB et SVM, avec le langage R.
La démarche présentée porte sur l'algorithme RF. Toute différence significative entre deux algorithmes, dans l'approche adoptée ou dans les résultats observés, sera spécifiquement mentionnée et détaillée.

Le protocole se décompose en 4 étapes :

i. *Tuning* des hyperparamètres (*cross-validation* et *grid search*) ;
ii. *Features selection* ;
iii. Méthodes de *subsampling* (*undersampling* et *oversampling*) ;
iv. Choix de la meilleure procédure.

II.5.a. Tuning des hyperparamètres (cross-validation et grid search)

Définition : L'optimisation des hyperparamètres (ou *tuning* en anglais) correspond à la recherche des valeurs d'hyperparamètres générant les meilleures métriques de précision.

Comme précisé, chaque algorithme possède plusieurs hyperparamètres, la recherche manuelle d'un optimum dépendant de plusieurs facteurs s'avère rapidement fastidieuse et peu concluante. D'où la nécessité d'utiliser une méthode robuste : la validation croisée.

Définition : La validation croisée (ou *cross-validation*) permet d'entraîner et de tester l'algorithme sur différentes segmentations de la base d'apprentissage, afin de réduire le biais et la variance associés à un apprentissage sur une unique partition arbitraire de la base d'apprentissage.

La base de données étudiée doit permettre deux choses :

- Entraîner l'algorithme de ML (phase d'apprentissage) ;
- Tester l'algorithme de ML (phase de test).

Réutiliser les mêmes données à la fois pour l'apprentissage et pour le test de l'algorithme ne permet pas d'étudier objectivement le comportement de la méthode face à un cas réel. Cet enjeu est primordial, car le *Machine Learning* intervient spécifiquement pour conjecturer des résultats à partir de données non rencontrées par la fonction prédictive (on parle alors de « généralisation de la méthode »). Il faut par conséquent éviter que le modèle ne soit trop spécialisé sur les données de la base d'apprentissage (cas d'*overfitting*), ou bien qu'il ne le soit pas assez et donc incapable de fournir des prédictions fiables (situation d'*underfitting*).

En outre, il serait fâcheux d'employer l'intégralité de la base de données pour l'apprentissage de l'algorithme, parce qu'il ne resterait plus aucune donnée pour le tester.

Enfin, la validation croisée requiert nécessairement plus de temps puisque le modèle sera entraîné et testé plusieurs fois.

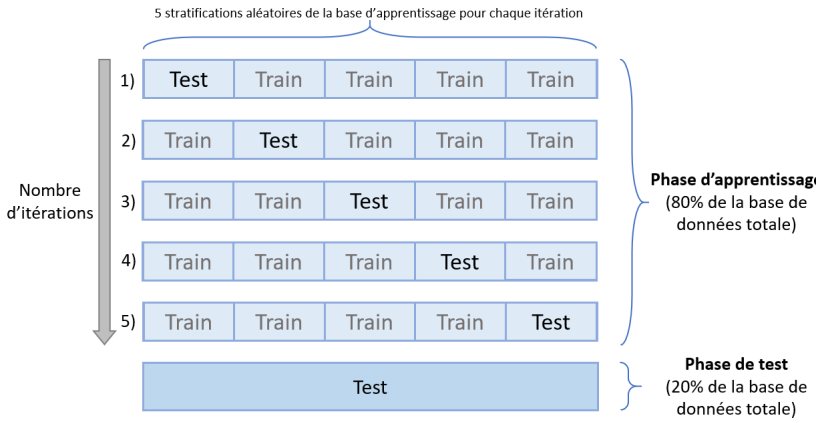

Figure 8. Mécanisme d'une 5-Fold Cross Validation

La méthode retenue pour cette étude est la *K-Fold Cross-Validation*. Celle-ci consiste en une stratification aléatoire de la base en K sous-échantillons de mêmes effectifs : le modèle est successivement entrainé sur les différentes combinaisons de K-1 stratifications de la base et testé sur la dernière subdivision. Cette approche est considérée comme performante parce qu'elle augmente la robustesse du modèle en réalisant des entrainements et des tests sur de nombreux sous-échantillons. L'enjeu principal d'une validation croisée avec la méthode *K-Fold* est par conséquent de déterminer le nombre K de subdivisions. Un équilibre doit être trouvé : un K élevé implique certes de nombreuses partitions, mais diminue tout autant la taille de l'échantillon test.
En l'absence de validation croisée, l'usage est d'entrainer l'algorithme sur 80% de la base et de le tester sur les 20% restants. Cette intuition puise notamment son inspiration dans le principe empirique de Pareto, selon lequel 80% des effets sont le produit de 20% des causes. En référence à cette règle heuristique, le coefficient retenu ici pour la *K-Fold Cross-Validation* est de 5.

Définition : *Grid Search* est une procédure dont le but est de déterminer méthodiquement la meilleure combinaison d'hyperparamètres pour optimiser une métrique de précision d'un modèle, tout en minimisant le nombre de simulations effectuées.

Pour cette étude, afin de tenir compte de l'*expected accuracy*, la métrique optimisée est le Kappa.

Les méthodes aléatoires ont été présentées comme offrant de bons résultats relativement aux méthodes manuelles (*Manual Grid Search*) dans plusieurs publications (voir Bergstra, J. et Bengio, Y., (2012), *Random Search for Hyper-Parameter Optimization*).

La méthode aléatoire utilisée est le *Random Latin Hypercube Grid Search*, introduite dans les travaux de McKay, M.D. et al., (2000), *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code*. Celle-ci fait appel à la méthode d'échantillonnage par hypercube latin, qui permet de répartir un échantillon dans un espace délimité à plusieurs dimensions sans que deux individus n'aient de coordonnées en commun. En d'autres termes, l'hypercube doit comporter un même nombre de coordonnées sur chaque intervalle, qui sont tirées aléatoirement et sans remise, afin de générer des individus positionnés stratégiquement dans cet espace orthogonal borné.

Pour un modèle avec n hyperparamètres (hypercube à n dimensions), le processus est composé des cinq étapes suivantes :

i. Les intervalles de l'hypercube sont d'abord déterminés pour les hyperparamètres : ils délimitent les valeurs des hyperparamètres qui seront balayées par le *grid search* ;
ii. Les hyperparamètres entiers sont traités en priorité. Le nombre k de valeurs équidistantes à tester est fixé : l'intervalle est découpé en k-1 segments égaux et les délimitations de ces segments seront les k valeurs des hyperparamètres qui seront testées par la méthode *grid search* ;
iii. Les hyperparamètres continus sont traités dans un second temps. L'intervalle est aussi découpé en k segments égaux. La différence fondamentale avec les hyperparamètres entiers réside dans le fait qu'une loi uniforme permettra de tirer aléatoirement une valeur pour chaque segment ;
iv. La métrique de précision est calculée pour différentes combinaisons d'hyperparamètres tirées aléatoirement parmi les valeurs fixées aux étapes précédentes, sans remise (i.e. chaque valeur n'est tirée qu'une seule fois) ;
v. Les hyperparamètres optimaux sont déterminés comme ceux maximisant la métrique de précision retenue.

Les intervalles sont aisément modulables au gré de l'utilisateur, qui peut choisir de les redimensionner pour rendre plus cohérent leur découpage. Par exemple dans le cas du modèle RF présenté ci-après, la variable *ntree* est considérée comme discrète afin de balayer des valeurs espacées de 200.

Le modèle RF comportant deux hyperparamètres, le *grid search* porte sur un hypercube de dimension 2 (i.e. un carré), ce qui rend l'approche plus intuitive :

(i) L'hyperparamètre discret *mtry* prend ses valeurs dans les entiers de l'intervalle borné [1 ; 9], ce qui délimite une arrête du carré. Une arrête allant de 100 à 1700 est délimitée pour l'hyperparamètre *ntree*.

(ii) RF ne comportant qu'un seul hyperparamètre discret et borné, la *subdivision* est fixée à 8 (9 valeurs délimitant 8 intervalles).

(iii) Dans un souci de cohérence avec la variable *mtry*, 8 intervalles avec un pas de 200 sont délimités pour la variable *ntree*. Celle-ci est donc considérée comme une variable discrète pour ce cas spécifique.

(iv) Neuf couples d'hyperparamètres sont tirés aléatoirement et sans remise parmi les jeux fixés aux étapes précédentes :

```
[1] "Urne: "
[1]  100  300  500  700  900 1100 1300 1500 1700
[1] "Tirage: 1100"
[1] "Urne: "
[1] 1 2 3 4 5 6 7 8 9
[1] "Tirage: 5"
[1] "Urne: "
[1]  100  300  500  700  900 1300 1500 1700
[1] "Tirage: 900"
[1] "Urne: "
[1] 1 2 3 4 6 7 8 9
[1] "Tirage: 7"
[1] "Urne: "
[1]  100  300  500  700 1300 1500 1700
[1] "Tirage: 300"
[1] "Urne: "
[1] 1 2 3 4 6 8 9
[1] "Tirage: 1"
[1] "Urne: "
[1]  100  500  700 1300 1500 1700
[1] "Tirage: 1500"
[1] "Urne: "
[1] 2 3 4 6 8 9
[1] "Tirage: 2"
[1] "Urne: "
[1]  100  500  700 1300 1700
[1] "Tirage: 700"
[1] "Urne: "
[1] 3 4 6 8 9
[1] "Tirage: 4"
[1] "Urne: "
[1]  100  500 1300 1700
[1] "Tirage: 1700"
[1] "Urne: "
[1] 3 6 8 9
[1] "Tirage: 9"
[1] "Urne: "
[1]  100  500 1300
[1] "Tirage: 1300"
[1] "Urne: "
[1] 3 6 8
[1] "Tirage: 8"
[1] "Urne: "
[1] 100 500
[1] "Tirage: 100"
[1] "Urne: "
[1] 3 6
[1] "Tirage: 6"
  ntree mtry
1  1100    5
2   900    7
3   300    1
4  1500    2
5   700    4
6  1700    9
7  1300    8
8   100    6
9   500    3
```

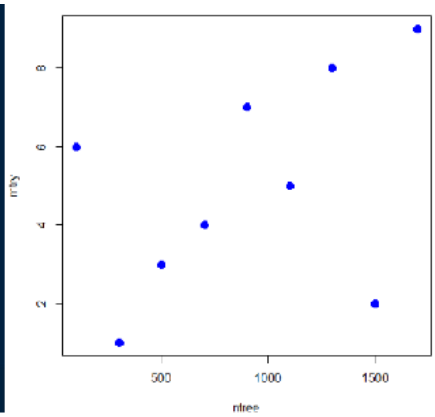

Figure 9. Tirage aléatoire selon *Random Latin Hypercube*

(v) Le couple d'hyperparamètres optimal selon *Kappa* est (mtry* = 8 ; ntree* = 1300). Le gain par rapport à la valeur minimale tirée dans l'échantillon est conséquent :

Nb de couples testés : 9	Kappa	*Balanced accuracy*
Valeur min pour (1 ; 300)	53,7%	43,3%
Valeur max pour (8 ; 1300)	86,8%	89,1%

Logiciel : R / Modèle : RF

Figure 10. Résultats pour *Random Latin Hypercube Grid Search*

II.5.b. Features selection

Définition : La sélection de variables (plus communément appelée *Features Selection*) est un procédé permettant de sélectionner un sous-ensemble de variables parmi celles composant la base d'apprentissage, selon leur contribution à la précision du modèle, et d'écarter les variables moins pertinentes.

Les principales cibles de la sélection de variables sont les catégories de données redondantes ou superflues. Le premier cas correspond aux catégories dont la valeur ajoutée est déjà perceptible par l'intermédiaire d'une autre variable (par exemple, quand deux variables sont fortement corrélées), le second est composé des catégories pour lesquelles la valeur ajoutée est jugée non-pertinente pour l'étude.

Le procédé de *Features Selection* permet une simplification des modèles (notamment quant à leur interprétation), une diminution du temps de calcul et une réduction du risque d'*overtfitting* (permettant ainsi une plus grande probabilité de généraliser le modèle).
La contribution des variables est traduite par le calcul de la *Mean Decrease Accuracy* (ou MDA). La MDA est calculée pour chaque variable : les modalités sont remplacées aléatoirement par des modalités selon une distribution équivalente, et la précision du modèle (*accuracy* présentée plus haut) est recalculée avec cette nouvelle base. Pour une variable peu ou pas importante, la permutation aura un impact non significatif sur la précision du modèle, à l'inverse des variables dont la contribution est importante.

Les différentes contributions sont ensuite triées sur un graphique selon leur MDA : les variables avec les MDA les plus faibles peuvent ainsi être retirées de la base d'apprentissage pour tenter d'améliorer les performances du modèle.
Il est à noter que l'attention doit être portée au classement des différentes variables et non à l'amplitude des mesures.

Analyse
La classification des contributions des variables selon leur MDA permet de constater une rupture (aussi appelée *coude*) au niveau de la variable *Statut matrimonial*. Même si l'explication des contributions de chaque variable n'est pas le sujet de ce papier, cette observation traduit le fait que la valeur ajoutée (en termes d'information) comprise dans la catégorie *Statut matrimonial* est négligeable.

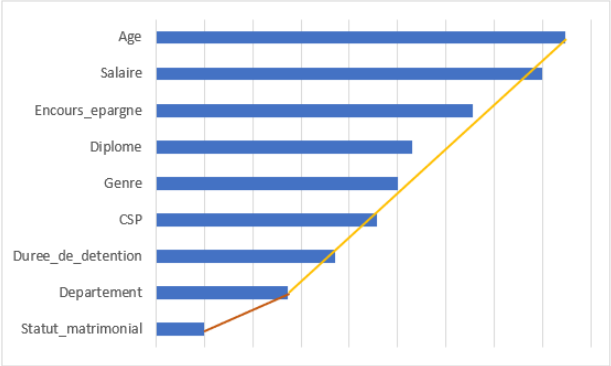

Figure 11. *Mean Decrease Accuracy* pour RF

La variable est par conséquent retirée de la base pour la suite de l'étude :

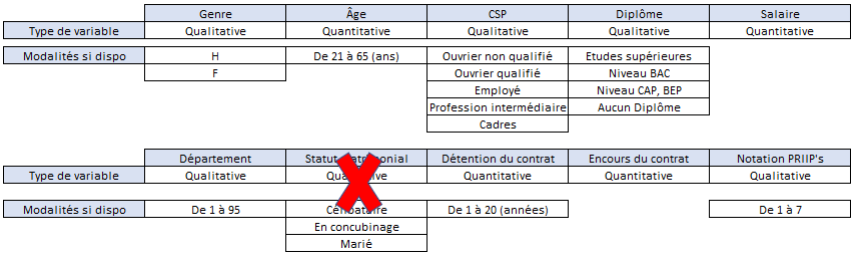

Figure 12. Retrait de la variable *Statut matrimonial* sur le modèle RF

La sélection de variables contribue aux impacts suivants :

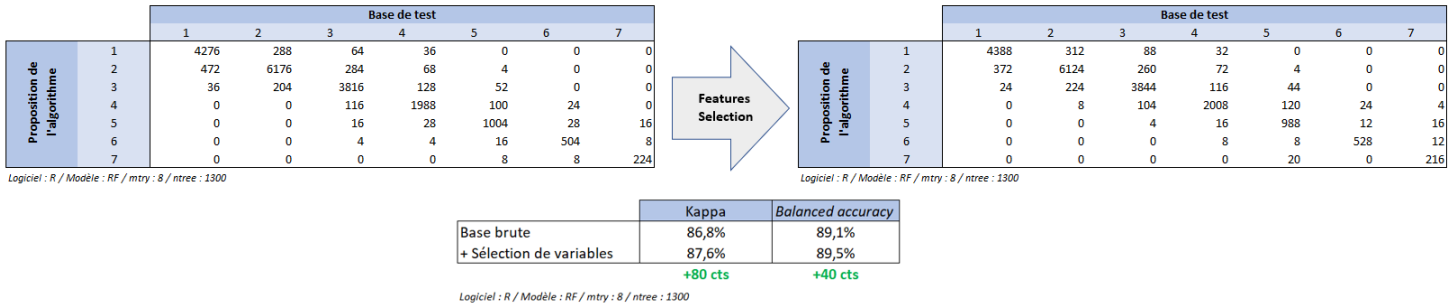

Figure 13. Sélection des variables pour l'algorithme RF

Après un *tuning* des hyperparamètres, une légère amélioration des métriques est observée : +40 cts sur la *balanced accuracy* et +80 cts sur le Kappa. En outre, les hyperparamètres optimaux *mtry* et *ntree* sont identiques.

Même si le gain sur les métriques de précision peut sembler léger, la sélection de variables permet surtout de diminuer la dimension de la base de données, permettant ainsi de réduire le temps de calcul (phénomène quantifiable pour des bases plus volumineuses) et de simplifier l'interprétation des résultats (non traitée dans cette contribution).

Une fois la variable avec la contribution la plus faible retirée, il convient de recalculer les contributions des variables restantes selon leur MDA et de réaliser à nouveau le processus. Dans le cas de cette étude, le retrait d'une variable supplémentaire dégrade les métriques de précision, donc seule la variable *Statut matrimonial* est retirée pour la suite de l'étude.

II.5.c. Méthodes de subsampling (undersampling et oversampling)

Comme précisé dans le paragraphe consacré à la *balanced accuracy*, un enjeu de cette étude est de détecter équitablement les individus dont le couple (profil de risque ; investissement) n'est pas statistiquement cohérent. Cette détection ne doit pas être influencée par la prépondérance de certaines catégories, pour lesquelles la précision et la fiabilité des résultats seraient améliorées au détriment des catégories sous-pondérées. Dans certains cas, la phase de *tuning*, en maximisant des métriques biaisées par la prépondérance de certaines classes (comme l'*accuracy*), peut aboutir à un hyperparamétrage optimal sur l'ensemble des données, aux dépens de la précision des labels sous-représentés, alors que ces derniers sont la cible de l'étude.

Ici les catégories les plus rares, à savoir les notations PRIIP's 5, 6 et 7, représentent justement les sous-populations les plus intéressantes pour l'assureur : mieux détecter ces types de profil augmentera le succès de la campagne commerciale.

Définition : Le *subsampling* est une méthode permettant de compenser un phénomène de déséquilibre au sein d'une base de données multi-classes. Elle permet de créer une base de données équitablement répartie entre les labels, normalisant ainsi la prise en compte de toutes les classes dans la phase d'apprentissage. Les deux méthodes de *subsampling* testées sont l'*undersampling* et l'*oversampling*.

Définition : Une méthode d'*undersampling* écarte aléatoirement des individus de la base parmi les classes les plus représentées, jusqu'à ce que chaque classe ait le même nombre d'individus que la classe initialement la moins représentée.

Définition : Une méthode d'*oversampling* complète les classes les moins représentées en renforçant leurs effectifs par des copies d'individus présents dans ces classes, jusqu'à atteindre une base équilibrée.

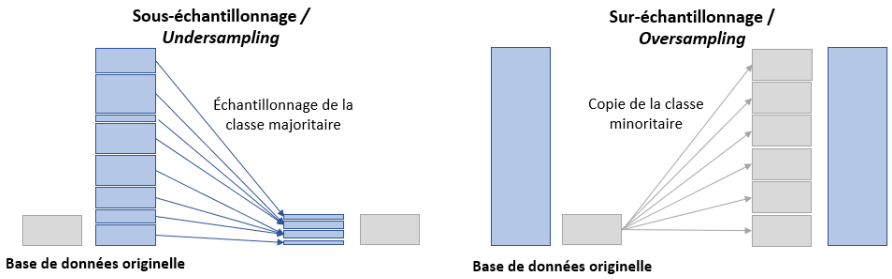

Figure 14. Principes des méthodes de *subsampling*

Il est à noter que l'objectif de cette mesure n'est pas d'équilibrer la base de test : celle-ci doit toujours être répartie telle qu'elle serait observée dans un cas réel.

De plus, les deux méthodes de *subsampling* doivent être utilisées de façon consistante avec une *cross-validation*. En effet, si la base de données est retraitée par une méthode de *subsampling* en amont d'un *tuning* des hyperparamètres par *cross-validation*, les *folds* générés seront déterminés à partir d'une base de données tributaire du *subsampling* effectué (un tirage différent de *subsampling* pourrait donner un résultat totalement différent). De même, le modèle risque d'apprendre sur des données non représentatives d'un cas réel, biaisant ainsi les résultats.

Aucune de ces deux méthodes ne peut être présentée comme étant plus efficace que l'autre : chacune devra être étudiée afin de déterminer laquelle correspond le mieux à la base de données et au modèle de ML utilisés, à travers les métriques de précision.

II.5.d. Choix de la meilleure procédure

Procédure utilisée	Kappa	*Balanced accuracy*
sans *subsampling*	87,6%	89,5%
avec *undersampling*	71,1%	81,5%
Dégradation des deux métriques		
avec o*versampling*	87,2%	89,3%
Métriques stables		
+ amélioration de la précision pour notation PRIIP's 7		

Logiciel : R / Modèle : RF / mtry : 8 / ntree : 1300

Figure 15. Métriques de précision obtenues par *subsampling* pour RF

Pour les deux méthodes de *subsampling*, les résultats sont significativement différents.
La méthode d'*undersampling* dégrade sensiblement les métriques de précision (Kappa et *balanced accuracy*) : cette méthode ne peut pas être considérée comme adaptée à l'étude réalisée.

La méthode d'*oversampling* dégrade très légèrement les deux métriques. Néanmoins, elle remplit son rôle : la précision sur la notation PRIIP's 7, qui représente seulement 1% de la base de d'apprentissage, passe de 87% à 90%. Ce gain prend toute son importance si le choix est fait de cibler spécifiquement les individus susceptibles d'être intéressés par cette catégorie d'investissement.

II.6. Différentes approches selon l'algorithme

II.6.a. Gradient Boosting : méthodes grid search alternatives

Le protocole présenté dans le cadre du modèle RF a aussi été appliqué au modèle GB.

Ce modèle comportant six hyperparamètres, la pertinence d'un *random latin hypercube grid search* est remise en question. En effet, les méthodes de type *random search* peuvent perdre en efficacité avec l'accroissement du nombre de dimensions, à l'inverse des méthodes dites "manuelles et séquentielles" (*sequential manual grid search*) qui, si elles sont encadrées par un expert, peuvent offrir de meilleures performances (voir Larochelle, H. et al., (2007), *Exploring strategies for training deep neural networks*).

Ainsi, pour le modèle GB, trois méthodes de *grid search* ont été comparées :

- *random latin hypercube grid search*, présentée en II.5.a. ;
- *random search*, qui sélectionne la combinaison d'hyperparamètres optimale parmi une sous-population générée aléatoirement et avec remise ;
- *sequential manual grid search*, qui consiste en une approche déterministe et non aléatoire.

La méthode *sequential manual grid search* est issue d'une publication sur le site Kaggle intitulée *Visual XGBoost Tuning with caret* (2018). Cette approche choisit de ne pas faire appel à un processus aléatoire et de prioriser certains hyperparamètres, à travers une méthode manuelle et séquentielle en 5 étapes :

i. Attribution de valeurs pour le taux d'apprentissage *eta* et le nombre d'itérations *nrounds* ;
ii. Recherche des hyperparamètres optimaux de profondeur *max depth* et de contrôle du nombre minimal d'observations dans un nœud terminal *min child weight* ;
iii. Cadrage de *colsample bytree* et de *subsample* ;
iv. Test de plusieurs valeurs pour *gamma* ;
v. Réduction du taux d'apprentissage *eta*.

Dix phases de *tuning* ont été menées à bien respectivement avec les méthodes *random latin hypercube grid search* et *random search* afin de pouvoir comparer les moyennes des résultats obtenus sur ces simulations avec le résultat de la méthode *sequential manual grid search* :

Moyenne sur 10 tirages	Kappa	*Balanced accuracy*
Random	85,2%	86,2%
Random Latin Hypercube	85,9%	87,4%
Tirage unique		
Sequential Manual	87,8%	87,9%

Logiciel : R / Modèle : GB

Figure 16. Comparaison des résultats pour GB selon la méthode de *grid search*

La méthode *sequential manual grid search* offre une plus grande facilité d'implémentation et de meilleurs résultats pour le modèle GB : son utilisation est recommandée au sein du protocole présenté dans cette contribution.
Le protocole de calibrage est complété avec une sélection de variables ainsi qu'une méthode de *subsampling* (*oversampling*), et permet d'obtenir les résultats suivants :

Métriques post-protocole	Kappa	*Balanced accuracy*
avec *oversampling*	87,4%	88,4%

Logiciel : R / Modèle : GB / eta : 0,025 / col = 0.8 / max_d : 7 / sub_s : 1 / γ = 0,7 / min_child : 1 / nro : 1200

Figure 17. Résultats post-protocole pour GB

II.6.b. Support Vector Machine : multi-classification et limites du modèle

Le *tuning* des hyperparamètres, réalisé à l'aide d'un *random latin hypercube grid search*, génère les résultats suivants :

		Base de test						
		1	2	3	4	5	6	7
Proposition de l'algorithme	1	3836	640	80	52	0	0	0
	2	928	5764	972	60	8	0	0
	3	20	244	2984	332	56	4	0
	4	0	20	248	1660	224	36	8
	5	0	0	12	140	804	116	68
	6	0	0	4	8	92	408	112
	7	0	0	0	0	0	0	60

Logiciel : R / Modèle : SVM / c : 1 / γ : 0,062

Support Vector Machine	Kappa	*Balanced accuracy*
avec *tuning* des hyperparamètres	70,5%	67,7%

Logiciel : R / Modèle : SVM / c : 1 / γ : 0,062

Random Forest	Kappa	*Balanced accuracy*
avec *tuning* des hyperparamètres	86,8%	89,1%

Logiciel : R / Modèle : RF / mtry : 8 / ntree : 1300

Gradient Boosting	Kappa	*Balanced accuracy*
avec *tuning* des hyperparamètres	87,8%	87,9%

Logiciel : R / Modèle : GB / eta : 0,025 / col = 0.8 / max_d : 7 / sub_s : 1 / γ = 0,7 / min_child : 1 / nro : 1200

Figure 18. Matrice de confusion pour SVM et comparaison des métriques avec les autres modèles en phase de *tuning*

Dès la phase de *tuning*, les métriques obtenues pour SVM sont sensiblement inférieures à celles des modèles RF et GB.

Les méthodes SVM furent initialement proposées pour réaliser de la classification binaire (Vapnik, P. et Cortes C., (1995), *Support-Vector Networks*). Le modèle SVM cherchant à maximiser les distances entre la frontière tracée et les deux groupes de données qu'elle sépare, le problème peut être contourné par une méthode itérative. L'approche "*one vs all"* propose d'abord de

séparer une classe spécifique du reste des données (les données restantes étant alors considérées comme la seconde classe) et de réaliser un apprentissage du modèle SVM dans cette configuration. Ensuite, ce procédé est répété itérativement, jusqu'à disposer d'un classifieur pour chaque classe.

Cette approche est décrite par Ahuja, Y et Yadav, S.K., (2012) dans *Multiclass classification and Support Vector Machine* comme étant fastidieuse et souvent utilisée par défaut, la séparation des données dans l'espace par un hyperplan atteignant rapidement ses limites dans le cas d'une classification non binaire.

À ce sujet, Pal, M. et Mather, P.M., (2005) décrivent dans *Support vector machines for classification in remote sensing* la faible robustesse du modèle SVM pour classifier des données avec des labels prépondérants, comme dans le cas de cette étude avec les classes 1, 2 et 3 qui sont labélisées pour 80% de la base d'apprentissage.

Malgré l'utilisation de méthodes de sélection de variables et de *subsampling*, le protocole ne permet pas d'améliorer suffisamment les métriques de précision du modèle SVM pour pouvoir concurrencer RF et GB :

Métriques post-protocole	Kappa	*Balanced accuracy*
avec *oversampling*	75,6%	78,5%

Logiciel : R / Modèle : SVM / c : 1 / γ : 0,062

Figure 19. Résultats post-protocole pour SVM

III. Résultats

III.1. Protocole appliqué à *Random Forest*

	Kappa	*Balanced accuracy*
Hyperparamétrage suboptimal	53,7%	43,3%
Tuning avec Grid Search ↓	**Gain important sur les métriques**	
Hyperparamétrage optimal	86,8%	89,1%
Sélection de variables ↓	**Léger gain sur les métriques & diminution du temps de calcul**	
Avec sélection de variables	87,6%	89,5%
Subsampling ↓	**Gain sur la précision de la classe 7**	
Avec *oversampling*	87,2%	89,3%

Figure 20. Protocole de calibrage appliqué au modèle RF

III.2. Performances des modèles post-protocole

	Random Forest		***Gradient Boosting***		***Support Vector Machine***	
	Kappa	*Balanced accuracy*	Kappa	*Balanced accuracy*	Kappa	*Balanced accuracy*
Résultats post-protocole	87,2%	89,3%	87,4%	88,4%	75,6%	78,5%

Figure 21. Résultats post-protocole pour les modèles RF, GB et SVM

IV. Discussion

IV.1. Résumé des analyses

La **Fig. 20** détaille les impacts des différentes étapes du protocole de paramétrage :

- L'étape de *tuning* avec une méthode de type *random latin hypercube grid search* génère un gain très important sur les métriques de précision ;

- La sélection de variables ne va pas forcément améliorer les métriques de précision mais va surtout diminuer la dimension de la base de données, permettant ainsi de simplifier l'interprétation des résultats et de réduire le temps de calcul. En outre, un phénomène non rencontré dans cet exemple peut aussi intervenir : la phase de *tuning* effectuée après la sélection de variables peut aboutir à un autre set d'hyperparamètres optimaux et encore diminuer le temps de calcul du modèle ;
- La méthode de *subsampling* retenue, à savoir l'*oversampling*, n'impacte que très légèrement les métriques de précision : son utilité se trouve dans l'amélioration de la détection de la catégorie la moins représentée.

La **Fig. 21** présente les métriques de précision post-protocole pour les trois modèles étudiés.

Random Forest et *Gradient Boosting* génèrent des métriques sensiblement similaires pour cette étude. *Random Forest* offrirait en temps normal un paramétrage plus intuitif, avec deux hyperparamètres contre sept pour le modèle *Gradient Boosting*. Néanmoins, la complexité de la phase de *tuning* pour le modèle GB a été contournée à l'aide d'un *sequential manual grid search*, ce qui place ces deux modèles sur un pied d'égalité pour cette étude.

Support Vector Machine produit de moins bons résultats. Ce modèle a été initialement introduit pour réaliser de la classification binaire, et sa faible robustesse dans le cas d'une classification avec un nombre de labels supérieur à deux a déjà observée. Le protocole proposé ne parvient pas à compenser ce phénomène, inhérent au modèle SVM.

IV.2. Travaux futurs

Tout d'abord, cette étude pourrait être complétée avec l'analyse d'autres algorithmes de *Machine Learning*, comme le modèle *Neural Networks*.

De plus, la loi PACTE (Plan d'Action pour la Croissance et la Transformation des Entreprises) promulguée en 2019 soulève de nouvelles problématiques. En effet, elle permet à n'importe quel assuré de transférer sans frais son épargne, alors répartie sur plusieurs contrats d'assurance vie d'un même assureur, au sein d'un unique contrat chez ce même assureur. Cette réglementation ouvre de nouvelles perspectives dans le cadre de la restructuration d'un passif d'assurance vie, des modèles de *Machine Learning* pouvant désormais être utilisées de façon croisée sur plusieurs bases (épargne, retraite, ...) avant de centraliser les résultats.

En outre, le protocole pourrait être complété en amont avec un processus de nettoyage des données, afin de corriger les différentes erreurs présentes dans une base de ce type (telles que les erreurs de frappe ou les données manquantes) et ainsi encore améliorer les performances des modèles de *Machine Learning*.

Enfin, une autre contribution peut être envisagée afin de couvrir tous les aspects d'une campagne de réallocation. Réalisée sur des données réelles, elle se concentrerait sur les différentes étapes du projet, de l'étude statistique jusqu'à l'implémentation en agence.

V. Conclusion

Face aux enjeux ALM auxquels les assureurs vie sont confrontés en période de taux durablement bas, la restructuration du passif s'impose comme solution et les méthodes de *Machine Learning* comme les outils nécessaires à son implémentation. La proportion restreinte d'assurés optant pour des allocations majoritairement investies en Unités de Compte, support plus rémunérateur pour l'assureur, limite la capacité d'un modèle de Machine Learning à apprendre à les détecter au sein d'un portefeuille d'assurés.

Le protocole proposé permet d'améliorer méthodiquement les performances d'un modèle de *Machine Learning* pour la classification d'une base non équilibrée, comme dans le cas de la restructuration du passif d'un assureur vie.

Dans le cas du modèle *Gradient Boosting*, l'utilisation lors de la phase de *tuning* d'un *sequential manual grid search* permet d'obtenir de meilleurs résultats qu'avec les méthodes aléatoires de type *random search* et *random latin hypercube grid search*.

Le modèle SVM n'apparaît pas comme un modèle pertinent à utiliser pour des problématiques de classification sur une base non équilibrée avec un nombre de classes supérieur à deux.

Références

II.5.a. *Tuning* des hyperparamètres (*cross-validation* et *grid search*)

- Bergstra, J. et Bengio, Y., (2012), *Random Search for Hyper-Parameter Optimization*, *Journal of Machine Learning Research 13*, 281-305
- McKay, M.D. et al., (2000), *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code*, Technometrics 42:1, 55-61

II.6.a. Gradient Boosting : méthodes *grid search* alternatives

- Larochelle, H. et al., (2007), *Exploring strategies for training deep neural networks*, *The Journal of Machine Learning Research*, 1-40
- Jani *[pseudonyme]*, (2018), *Visual XGBoost Tuning withcaret*, disponible sur www.kaggle.com/pelkoja/visual-xgboost-tuning-with-caret

II.6.b. Support Vector Machine : multi-classification et limites du modèle

- Vapnik, P. et Cortes C., (1995), *Support-Vector Networks*, *Mach Learn* 20, 273–297
- Ahuja, Y et Yadav, S.K., (2012), *Multiclass classification and Support Vector Machine*, *Global Journal of Computer Science and Technology Interdisciplinary*, Volume 12 Issue 11
- Pal, M. et Mather, P.M., (2005), *Support vector machines for classification in remote sensing*, *International Journal of Remote Sensing* 26:5, 1007-1011