Mémoire présenté devant l'ENSAE Paris
pour l'obtention du diplôme de la filière Actuariat
et l'admission à l'Institut des Actuaires
le 25 septembre 2024

Par : **Axel Merlin**

Titre : **Optimisation de l'allocation d'actifs cible en assurance-vie avec intelligence artificielle dans le cadre de Solvabilité 2**

Confidentialité : ☒ NON ☐ OUI (Durée : ☐ 1 an ☐ 2 ans)

Les signataires s'engagent à respecter la confidentialité indiquée ci-dessus

Membres présents du jury de la filière

Entreprise :

Nom : Cabinet Deloitte
Signature :

Membres présents du jury de l'Institut des Actuaires

Directeur du mémoire en entreprise :

Nom : Ning Lin
Signature :

Autorisation de publication et de mise en ligne sur un site de diffusion de documents actuariels *(après expiration de l'éventuel délai de confidentialité)*

Signature du responsable entreprise

Secrétariat :

Signature du candidat

Bibliothèque :

Axel Merlin

Ecole Nationale de la Statistique et de l'Administration Economique (ENSAE)
5, avenue Henry Le Chatelier - 91120 PALAISEAU, FRANCE

Remerciements

Je tiens à exprimer ma plus profonde gratitude envers toutes les personnes qui ont contribué à l'aboutissement de ce mémoire.

Tout d'abord, je souhaite remercier chaleureusement ma famille pour leur soutien indéfectible tout au long de mes études. Leur présence, leurs encouragements et leur confiance en moi ont été essentiels pour mener à bien ce projet.

Je voudrais également adresser mes sincères remerciements à monsieur Axel Elbaz, consultant chez Deloitte. Votre disponibilité et vos conseils avisés ont été d'une grande aide dans l'élaboration de ce mémoire. Merci pour votre encadrement et vos encouragements tout au long de cette période.

Je tiens aussi à remercier mon manager, monsieur Nin Ling, pour le sujet qui m'a été présenté.

Je souhaite exprimer toute ma reconnaissance à monsieur Baptiste Brechot, associé chez Deloitte chargé de l'équipe actuariat. Votre accompagnement m'a grandement influencé dans les orientations stratégiques de ce projet. Merci pour la confiance que vous m'avez accordée.

Je remercie également l'ensemble de l'équipe actuariat chez Deloitte. Votre collaboration, vos échanges constructifs et votre soutien tout au long de ce parcours ont été d'une grande valeur. Travailler à vos côtés a été une expérience enrichissante qui a fortement contribué à la qualité de ce mémoire.

Enfin, je tiens à remercier ChatGPT et, plus largement, l'IA générative, qui ont permis d'améliorer la productivité et la qualité de ce mémoire. Ces outils ont facilité l'organisation de mes idées et m'ont aidé à structurer mes pensées de manière claire et efficace. Leur contribution a été un atout précieux dans l'accomplissement de ce travail, notamment pour la rédaction de certaines et la présentation de certains principes actuariels et mathématiques.

À tous, un grand merci pour votre contribution à ce travail et pour votre soutien inestimable.

Table des matières

Résumé

Ce mémoire d'actuariat porte sur l'optimisation de l'allocation cible d'actifs en assurance-vie dans le cadre de Solvabilité II, en utilisant des techniques d'intelligence artificielle. L'étude commence par explorer le paysage de l'assurance-vie en France, en mettant en lumière les différents types de contrats. Elle aborde ensuite le contexte réglementaire, en particulier les trois piliers de Solvabilité II, avec un accent sur les exigences quantitatives (Pilier 1). Une partie importante de l'étude est consacrée à la mise au point de méthodes de recherche de l'allocation cible pour un portefeuille d'assurance vie.

Le développement du modèle d'optimisation a impliqué l'intégration de modèles d'apprentissage automatique tels que XGBoost, les réseaux de neurones et la régression à noyau. Ces modèles ont été formés pour prédire des net asset values (NAVs) à un environnement économique fixé. Des techniques d'optimisation avancées ont été utilisées pour explorer efficacement l'espace des allocations cibles. La mise en œuvre pratique comprenait la caractérisation des scénarios économiques, le développement d'interfaces de communication entre les différents outils et la sélection des entrées et des sorties appropriées. La méthodologie s'est concentrée sur l'amélioration de la précision des prédictions des allocations d'actifs cibles et la réduction du temps de calcul.

La conclusion de ce mémoire démontre que l'utilisation de l'intelligence artificielle permet d'aborder l'allocation cible d'actifs sous un nouvel angle. En proposant une méthode robuste pour identifier les portefeuilles cibles, ce mémoire offre un outil précieux aux assureurs pour explorer une nouvelle approche d'optimisation. L'étude montre que l'allocation cible optimisée n'est pas unique et que celle-ci doit être mise en perspective avec les multiples objectifs de l'assureur. En fournissant des solutions basées sur des techniques avancées de machine learning, ce mémoire contribue à la modernisation et à l'innovation dans le secteur de la gestion des actifs et des passifs, permettant une gestion désormais globale et non plus locale des stratégies d'allocation.

Abstract

This actuarial thesis focuses on the optimization of the target asset allocation in life insurance within the Solvency II framework, using artificial intelligence techniques. The study begins by exploring the landscape of life insurance in France, highlighting the different types of contracts. It then addresses the regulatory context, particularly the three pillars of Solvency II, with an emphasis on quantitative requirements (Pillar 1). A significant part of the study is dedicated to developing methods for determining the target allocation for a life insurance portfolio.

The development of the optimization model involved integrating machine learning models such as XGBoost, neural networks, and kernel regression. These models were trained to predict net asset values (NAVs) for a fixed economic environment. Advanced optimization techniques were used to efficiently explore the space of target allocations. The practical implementation included the characterization of economic scenarios, the development of communication interfaces between different tools, and the selection of appropriate inputs and outputs. The methodology focused on improving the accuracy of predictions for target asset allocations and reducing computation time.

The conclusion of this thesis demonstrates that the use of artificial intelligence allows for a new approach to target asset allocation. By proposing a robust method to identify target portfolios, this thesis offers a valuable tool for insurers to explore a novel optimization approach. The study shows that the optimized target allocation is not unique and must be considered in light of the insurer's multiple objectives. By providing solutions based on advanced machine learning techniques, this thesis contributes to the modernization and innovation in asset and liability management, enabling a now global rather than local approach to allocation strategies.

Note de synthèse

Ce mémoire propose une méthode pour permettre aux assureurs d'optimiser leur allocation cible d'actifs (obligation, action, immobilier, cash) en fonction d'indicateurs comme le ratio $\frac{\text{NAV}_{central}}{\text{SCR}_{marche}}$. L'approche proposée utilise les algorithmes de machine learning pour réduire les temps de calcul des simulations, mais aussi proposer une vision globale et non plus locale de l'allocation d'actifs.

Dans ce mémoire, l'univers économique est fixe, c'est-à-dire que les scénarios économiques proposés par le GSE sont fixes durant toute l'étude. Dans l'approche classique, pour obtenir une mesure de NAV pour un sous-module de risque, 1 000 NAVs stochastiques sont calculées à partir de 1 000 scénarios économiques. Ainsi, le procédé proposé se décompose en plusieurs étapes :

1. **L'échantillonnage des scénarios économiques :** En sélectionnant correctement les scénarios économiques, il est possible de réduire le nombre de scénarios économiques en entrée du modèle ALM en réduisant la perte d'information liée à l'échantillonnage.
2. **L'approximation du modèle ALM :** Le modèle ALM repose sur une méthode de Monte Carlo. Ce mémoire propose de remplacer le modèle ALM basé sur une méthode répétitive par des modèles de machine learning.
3. **L'algorithme de recherche du portefeuille optimal :** La recherche du portefeuille cible est une étape cruciale qui peut demander beaucoup de temps et de ressources. Il est crucial de proposer une méthode de convergence rapide avec de bonnes capacités exploratoires.

Pour échantillonner les scénarios économiques, composés des taux forward pour différents indicateurs économiques (3 264 variables), il est nécessaire de choisir une méthode pour réduire la dimension des scénarios économiques et ensuite les échantillonner. Un autoencodeur avec une erreur quadratique moyenne de reconstruction de 0.00887 a donc été utilisé. Enfin, un échantillonnage par maximisation de la diversité est utilisé pour sélectionner les scénarios économiques. La base d'entrainement est ainsi constituée.

Pour approximer le modèle ALM, des fonctions d'approximation sont développées pour prédire la NAV en fonction de l'allocation d'actifs cible selon les univers de risques suivants :

— central
— risque action
— risque immobilier
— risque spread
— risque taux à la hausse
— risque taux à la baisse

Pour construire les multiples fonctions d'approximation, différents types d'algorithmes et de stratégies d'entrainement sont utilisés. Les algorithmes suivants ont été étudiés :

— XGBoost
— Dense neural network (DNN)
— Kernel ridge régression

L'étude de différents modèles permet d'identifier l'approche la plus adaptée à la nature des données et à la tâche à accomplir. Chaque modèle repose sur des principes distincts pour capturer les relations présentes dans les données, offrant ainsi des perspectives complémentaires.

En comparant ces modèles, la solution finale maximise la robustesse et les performances. Cette diversité d'approches permet de sélectionner le modèle le plus performant et généralisable pour le problème étudié.

Ensuite, chaque modèle a été entrainé en utilisant l'ensemble de la base d'apprentissage ou bien en étant entrainé sur un sous-module de risques spécifiques. Les résultats ont montré que le modèle final est le suivant :

- Le modèle XGBoost entrainé sur l'ensemble de la base de donnée prédit les NAVs pour le scénario central.
- Les modèles XGBoost entrainés sur des sous-modules de risque spécifiques prédisent les NAVs pour les risques action, immobilier, spread.
- Les régressions à noyaux avec pénalisation Ridge entrainées sur des sous-modules de risque spécifiques prédisent les NAVs pour les risques taux haut et taux bas.

L'erreur de prédiction sur les NAV est alors de l'ordre du pourcent.

Pour prédire le Solvency Capital Requirement (SCR), tel que défini par la norme Solvabilité II, les écarts entre la valeur nette des actifs (Net Asset Value, NAV) dans des conditions normales et la NAV après l'application de chocs prédéterminés sont agrégés suivant la formule standard.

Enfin, pour obtenir un modèle de recherche du portefeuille cible d'actifs, un Particle Swarm Optimizer (PSO) est utilisé. En travaillant sur le choix des paramètres, il est possible d'améliorer la convergence ou le caractère exploratoire de l'essaim. Pour améliorer davantage le PSO traditionnel, différentes populations de particules ont été mélangées :

- Particules standards
- Particules à mémoire
- Particules exploratoires
- Particules exploitatives
- Particules adaptives
- Particules quantiques

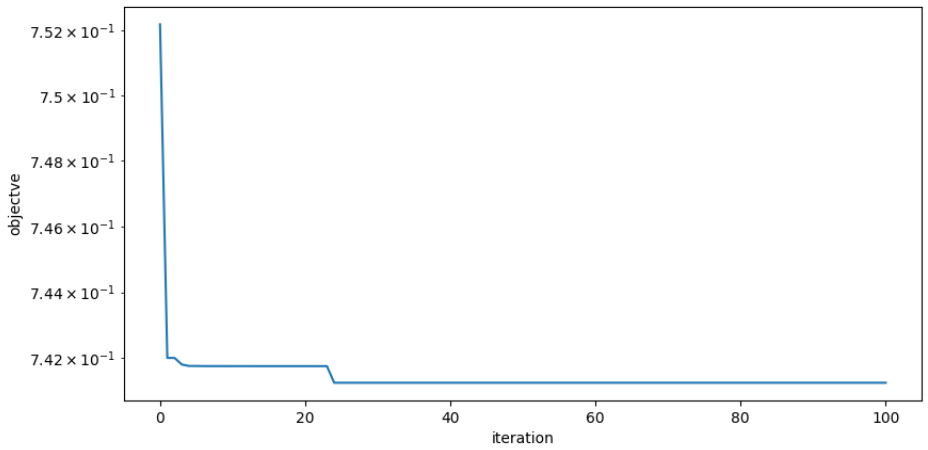

FIGURE 1 – Évolution de la fonction objectif en fonction des itérations pour l'algorithme PSO

Ce mémoire a cherché à optimiser dans un premier temps le ratio $\frac{\text{NAV}_{central}}{\text{SCR}_{marche}}$, mais un assureur ne cherche pas à optimiser un seul indicateur lorsque ce dernier décide de son allocation cible. En effet, pour répondre aux différentes normes auxquelles il est soumis et à ses objectifs, ce dernier doit pondérer sa fonction objective. À titre d'exemple, ce mémoire présente l'évolution de l'allocation cible x en fonction du paramètre κ.

$$\max_x NAV(x) - \kappa SCR(x)$$

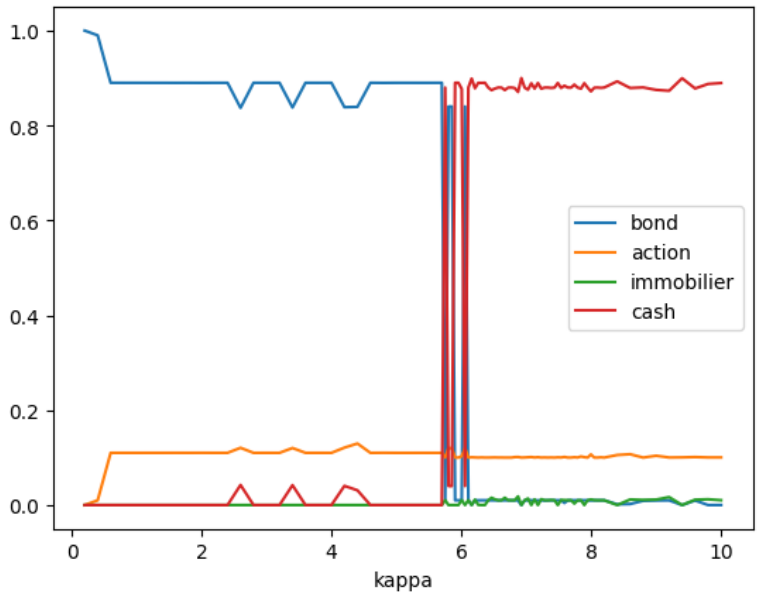

FIGURE 2 – Évolution du portefeuille cible en fonction du paramètre κ

Cette étude illustre l'impact de la fonction objectif sur l'allocation cible optimale. Le graphique 55 remet en question les idées reçues. En effet, pour minimiser le SCR, l'option la plus intuitive semble être une allocation cible de 100 % en cash. Cependant, le modèle proposé révèle que l'allocation optimale pour réduire le SCR est plutôt composée de 10 % d'actions et 90 % de cash. La figure 55 souligne également la transition entre différents portefeuilles en fonction de l'importance accordée au SCR ou la NAV en univers de risque central.

Une nouvelle approche pour mesurer la stabilité de la NAV en fonction de l'allocation cible est également proposée. Cette approche utilise le modèle ALM approximé par machine learning et l'apprentissage non supervisé pour identifier et quantifier les espaces à faible variation. Le portefeuille cible identifié 26 est très éloigné du portefeuille classique d'un assureur. Mais cette allocation est à contrebalancer avec la fonction objectif désirée : la stabilité de la NAV d'un assureur est une contrainte et non une cible à atteindre.

Actif	Part
Obligation	14,36 %
Action	16,30 %
Immobilier	16,65 %
Cash	46,31 %
Total	100 %

TABLE 1 – Portefeuille optimal au sens de la stabilité

Ainsi, la méthode proposée ne doit pas être restreinte à son cadre d'application, mais adaptée aux besoins des assureurs. Le procédé proposé permet d'obtenir, avec peu de ressources et de temps, une vision globale des portefeuilles cibles et non locale comme le proposent les traditionnels tests de sensibilités. L'ensemble du procédé établi se résume avec la figure 6.

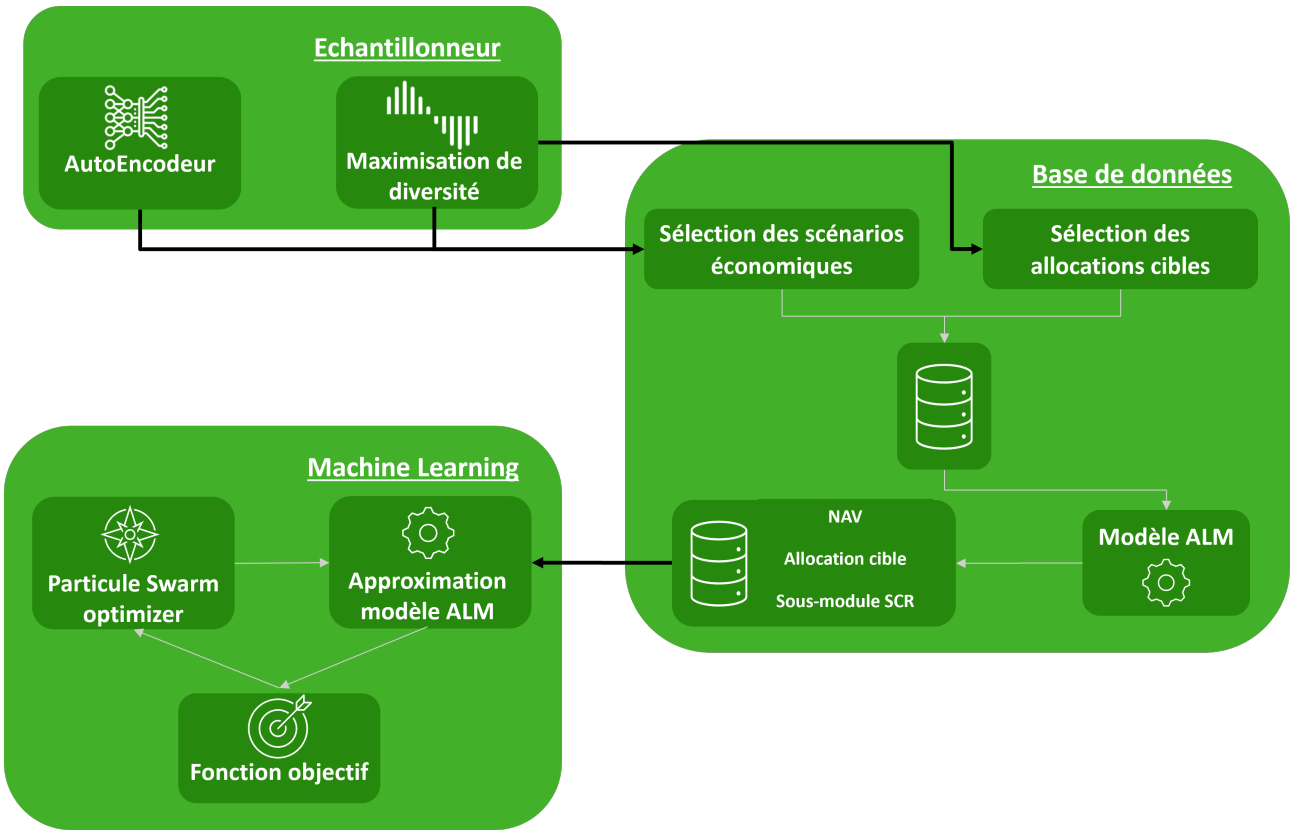

FIGURE 3 – Schéma récapitulatif du mémoire

Executive summary

This thesis proposes a method to enable insurers to optimize their target asset allocation (bond, equity, real estate, cash) based on indicators such as the $\frac{textNAV_{central}}{\text{SCR}_{market}}$ ratio. The proposed approach uses machine learning algorithms to reduce simulation calculation times, and also offers a global rather than local view of asset allocation.

In this paper, the economic environment is fixed. However, to obtain a measure of NAV for a risk sub-module, 1,000 simulations are required. Therefore, the proposed process is broken down into several steps :

1. **Sampling economic scenarios :** By correctly selecting economic scenarios, it is possible to reduce the number of economic scenarios input into the ALM model while minimizing the loss of information.
2. **Approximating the ALM model :** The ALM model is based on a Monte Carlo method. This paper proposes replacing the repetitive Monte Carlo-based ALM model with machine learning models.
3. **Searching for the optimal portfolio :** The search for the target portfolio is a crucial step that can require significant time and resources. It is essential to propose a method with fast convergence and strong exploratory capabilities.

To sample the economic scenarios, which consist of forward rates for various economic indicators (3,264 variables), a method for dimensionality reduction of the economic scenarios is necessary before sampling them. An autoencoder with a mean squared reconstruction error of 0.00887 was used. Finally, sampling by maximizing diversity was used to select the economic scenarios, thus constituting the training dataset.

To approximate the ALM model, approximation functions are developed to predict NAV as a function of target asset allocation according to the following risk universes :

— central
— equity risk
— real estate risk
— spread risk
— upward interest rate risk
— downward interest rate risk

To build the various approximation functions, different types of algorithms and training strategies were used. The following algorithms were studied :

— XGBoost
— Dense Neural Network (DNN)
— Kernel Ridge Regression

The study of different models allows for the identification of the most suitable approach for the nature of the data and the task at hand. Each model is based on distinct principles for capturing the relationships present in the data, thus offering complementary perspectives. By comparing these models, the final solution maximizes robustness and performance. This diversity of approaches enables the selection of the most efficient and generalizable model for the problem at hand.

Then, each model was trained using the entire training dataset or trained on specific risk sub-modules. The results showed that the final model is as follows :

— The XGBoost model trained on the entire dataset predict the NAVs for the Central scenario.
— The XGBoost models trained on specific risk sub-modules predict the NAVs for equity, real estate, and spread risks.
— Kernel Ridge Regression models trained on specific risk sub-modules predict the NAVs for upward and downward interest rate risks.

The prediction error on the NAV is then on the order of one percent.

To predict the Solvency Capital Requirement (SCR), as defined by the Solvency II standard, the differences between the Net Asset Value (NAV) under normal conditions and the NAV after the application of predetermined shocks are aggregated according to the standard formula.

Finally, to obtain a coherent target asset allocation model, a Particle Swarm Optimizer (PSO) is used. By fine-tuning the parameters, it is possible to improve the convergence or the exploratory nature of the swarm. To further enhance the traditional PSO, different populations of particles were mixed :

— Standard particles
— Memory particles
— Exploratory particles
— Exploitative particles
— Adaptive particles
— Quantum particles

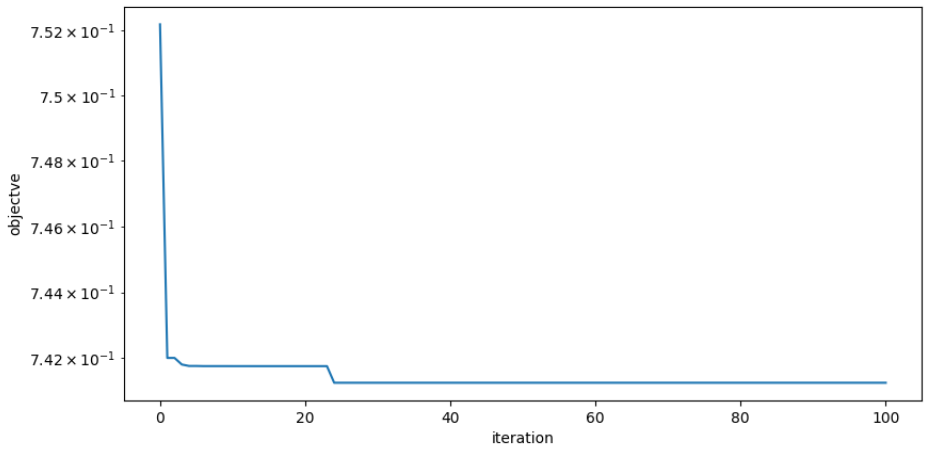

FIGURE 4 – Evolution of the objective function over iterations for the PSO algorithm

This paper initially sought to optimize the ratio $\frac{\text{NAV}_{central}}{\text{SCR}_{market}}$, but an insurer does not aim to optimize a single indicator when deciding on its target allocation. Indeed, to comply with various regulations and achieve its objectives, the insurer must weigh its objective function. As an example, this paper presents the evolution of the target allocation x as a function of the parameter κ.

$$\max_x NAV(x) - \kappa SCR(x)$$

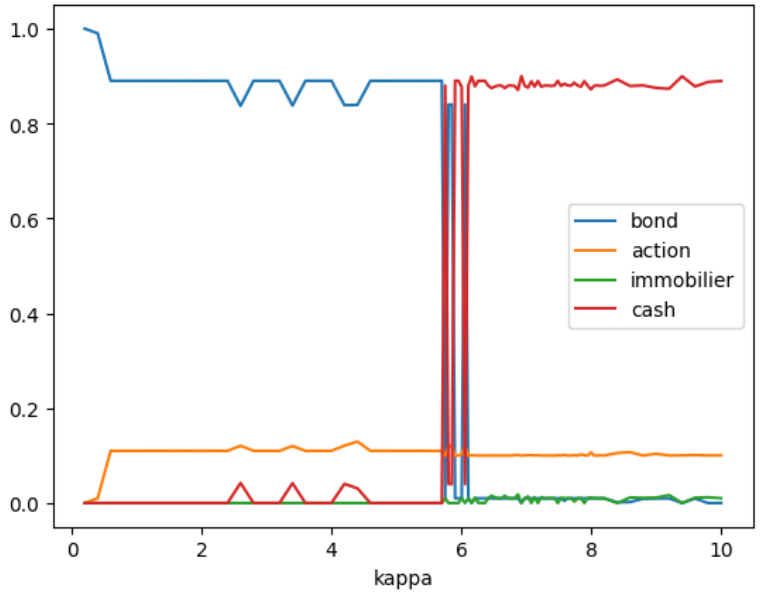

FIGURE 5 – Evolution of the target portfolio as a function of the parameter κ

This study illustrates the impact of the objective function on the optimal target allocation. The graph 55 challenges common assumptions. Indeed, to minimize the SCR, the most intuitive option seems to be a target allocation of 100% in cash. However, the proposed model reveals that the optimal allocation to reduce the SCR is instead composed of 10% in equities and 90% in cash. Figure 55 also highlights the transition between different portfolios depending on the emphasis placed on the SCR or the NAV in a central risk scenario.

A new approach to measure the stability of the NAV based on the target allocation is also proposed. This approach uses the ALM model approximated by machine learning and unsupervised learning to identify and quantify low-variation spaces. The identified target portfolio 26 is very different from a typical insurer's portfolio. However, this allocation must be balanced with the desired objective function : an insurer's NAV stability is a constraint, not a goal to achieve.

Asset	**Share**
Bonds	14.36 %
Equities	16.30 %
Real estate	16.65 %
Cash	46.31 %
Total	100 %

TABLE 2 – Optimal portfolio in terms of stability

Thus, the proposed method should not be limited to its application framework but adapted to the needs of insurers. The proposed process allows obtaining, with minimal resources and time, a global rather than a local view of target portfolios, as is often the case with traditional sensitivity tests. The entire established process is summarized in Figure 6.

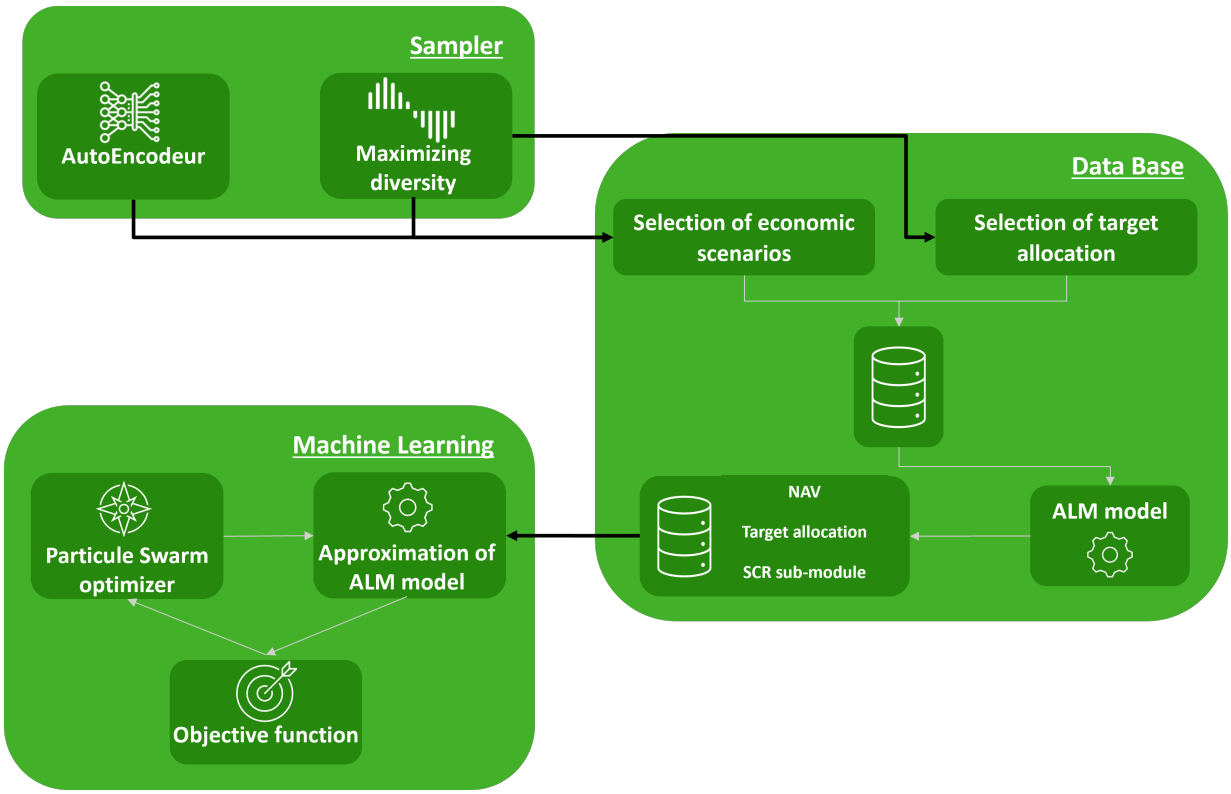

FIGURE 6 – Summary diagram of the process

Introduction

L'optimisation de l'allocation cible d'actifs est une préoccupation centrale pour les assureurs, qui doivent jongler entre la maximisation de la valeur nette des actifs (Net Asset Value, NAV) et le respect des exigences réglementaires, telles que celles imposées par Solvabilité II. Cette norme européenne impose aux compagnies d'assurance de maintenir un capital suffisant pour couvrir leurs risques, mesuré par le Solvency Capital Requirement (SCR). Pour répondre à ces exigences tout en optimisant leurs performances financières, les assureurs sont confrontés à des défis complexes qui nécessitent des approches innovantes.

Ce mémoire explore une méthode avancée pour optimiser l'allocation cible d'actifs d'un assureur en se focalisant dans un premier temps sur le ratio $\frac{\text{NAV}_{central}}{\text{SCR}_{marche}}$. L'objectif est de proposer une approche qui non seulement réduit les temps de calculs grâce à l'intégration d'algorithmes de machine learning, mais qui offre également une nouvelle perspective sur l'allocation d'actifs. Cette méthode repose sur l'utilisation d'un modèle ALM (Asset-Liability Management) simulant les différents scénarios économiques et leurs impacts sur les actifs et passifs de l'assureur. Afin de surmonter les limitations des méthodes traditionnelles, ce mémoire propose de remplacer certaines étapes du processus par des modèles de machine learning, permettant ainsi de réduire les besoins en calculs tout en améliorant la précision et la robustesse des résultats.

L'approche proposée s'articule autour de trois axes principaux : la réduction de la dimensionnalité des scénarios économiques, l'approximation du modèle ALM à l'aide d'algorithmes de machine learning, et l'optimisation du portefeuille d'actifs cible via des techniques d'optimisation avancées comme le Particle Swarm Optimizer (PSO). Chacune de ces étapes est cruciale pour atteindre une allocation d'actifs qui soit non seulement efficace mais aussi adaptable aux exigences réglementaires et aux objectifs stratégiques de l'assureur.

Ce mémoire vise à démontrer que l'intégration de techniques de machine learning dans le processus d'optimisation de l'allocation d'actifs permet non seulement de gagner en efficacité, mais aussi de proposer des solutions plus flexibles et adaptées aux besoins spécifiques des assureurs. L'objectif est de fournir une méthode qui puisse être facilement adaptée à différents contextes économiques et réglementaires, offrant ainsi une véritable valeur ajoutée aux processus décisionnels des compagnies d'assurance.

1 Présentation de l'assurance-vie

1.1 Définition de l'assurance-vie

Selon France Assureurs [Ass], en 2023, le secteur de l'assurance-vie en France a atteint des records avec un encours de 1923 milliards d'euros et une collecte nette de +2,4 milliards d'euros à la fin de l'année. La tendance semble se poursuivre sur l'année 2024, avec un niveau de cotisations en janvier 2024 atteignant 15,9 milliards d'euros, soit une augmentation de 12 % par rapport à l'année précédente. Cette croissance s'explique en partie par l'essor des Plans d'épargne retraite (PER) assurantiels, qui ont attiré de nombreux nouveaux assurés. Ce qui fait de l'assurance-vie le premier moyen d'épargne en France.

L'assurance-vie est un placement de long terme. Elle se décompose en trois types de contrats :
- l'assurance en cas de vie : Ce type de contrat garantit le versement d'un capital ou d'une rente à l'assuré s'il est en vie à l'échéance du contrat.
- l'assurance en cas de décès : Ce type de contrat garantit le versement d'un capital ou d'une rente aux bénéficiaires désignés en cas de décès de l'assuré avant l'échéance du contrat.
- l'assurance mixte vie et décès : Ce type de contrat combine les garanties des deux contrats précédents.

Les contrats doivent également être ventilés par le support d'investissement choisi :
- Les contrats souscrits en euros
- Les contrats souscrits en unités de compte ou actions
- Les contrats multisupports
- Les contrats Euro-croissance

Pour récupérer l'épargne investie, la plupart des contrats proposent trois options :
- Le rachat total de l'assurance-vie : fermeture définitive du contrat
- Le rachat partiel : récupération d'une partie du capital constitué
- La transformation en rente viagère : transforme l'épargne en revenu régulier

1.2 Les différents types de contrats

1.2.1 Contrat d'épargne en euros

Les contrats en fonds euros sont des produits d'assurance-vie particulièrement populaires en France en raison de leur sécurité. Ils se caractérisent par trois éléments :
- Capital garanti : L'assureur s'engage à rembourser au moins le montant initialement investi. Ainsi, peu importe les fluctuations du marché, l'assuré ne peut pas perdre de capital.
- Effet de cliquet : Les intérêts générés chaque année sont définitivement acquis et viennent s'ajouter au capital initial. Ainsi, chaque année, le capital dû à l'assuré augmente.
- Rendement : Le rendement des contrats en fonds euros est composé de deux éléments : un taux minimum garanti et une participation aux bénéfices. Cette structure permet de protéger les engagements de l'assureur tout en garantissant à l'assuré un revenu équitable.

Taux minimum garanti Le taux minimum garanti est un taux minimum de revalorisation garanti par l'assureur à l'assuré sur une période fixée. Ce taux peut être fixe ou basé sur un indice. Cependant, pour garantir la pérennité des activités de l'assureur, le Code des assurances

impose aux assureurs une borne supérieure à ce taux. Cette borne permet de limiter les engagements de l'assureur. D'après le Code des assurances (article A132-1) [Leg], le taux minimum ne peut pas excéder :

— 75 % du taux moyen des emprunts de l'État français (TME) pour un engagement inférieur à 8 ans (hors contrats à prime périodique ou à capital variable)
— Sinon, $\min(3,5\%, 60\%\text{TME})$

En décembre 2023, le TME en base semestrielle s'élève à 2,92%.

	Taux minimum garanti
Engagement ≤ 8 ans	2,19%
Sinon	1,75%

TABLE 3 – Taux minimum garanti appliqué

Participation aux bénéfices La participation aux bénéfices est également encadrée par l'article A132-11) [Leg] : L'assureur doit reverser à l'assuré au moins 90 % du résultat technique et 85 % du résultat financier. Ainsi, les assurés peuvent bénéficier des bonnes performances du marché. En revanche, l'assureur peut différer le versement de la participation aux bénéfices en constituant une provision aux bénéfices ou aux excédents. Par la constitution de cette provision, l'assureur se donne la possibilité de lisser son résultat.

Fonds Euros Les fonds Euros sont constitués de divers types d'actifs financiers pour assurer à la fois la stabilité et la sécurité des investissements. Les fonds euros sont majoritairement investis en obligations d'État et d'entreprise de haute qualité (Investment Grade). Ensuite, une petite proportion du fonds est investie en actions pour capter un potentiel de rendement supplémentaire tout en minimisant les risques. L'immobilier constitue également un axe d'investissement qui offre une diversification et des rendements stables. Enfin, une portion du fonds est maintenue en liquidité pour répondre aux besoins de liquidité immédiate et assurer la gestion du fonds.

1.2.2 Contrat d'épargne en unités de compte (UC)

Les contrats d'épargne en unités de compte sont des produits d'assurance-vie qui diffèrent des fonds euros par leur mode de fonctionnement et leur profil de risque :

— Absence de garantie du capital
— Potentiel rendement supérieur
— Diversification des investissements

Composition des unités de compte Les unités de compte sont marquées par une large diversification des investissements pour capter des rendements plus importants :

— Actions : pour capter le potentiel de croissance des entreprises, l'investissement peut être effectué dans des actions individuelles ou dans des fonds d'actions.
— Obligations : pour garantir des revenus fixes, l'investissement des unités de comptes peut inclure des obligations d'États, d'entreprises ou de fonds obligataires.
— Immobilier : les unités de comptes peuvent inclure des fonds immobiliers (SCPI, OPCI) qui investissent dans des actifs immobiliers pour assurer une diversification du portefeuille.
— Fonds multisupports : pour mieux répartir les risques, les contrats en unités de compte peuvent investir dans des fonds multisupports, qui sont diversifiés sur plusieurs types d'actifs.

La gestion des contrats en unités de compte est assurée par des professionnels qui sélectionnent les actifs en fonction des objectifs de rendement et de diversification. La performance des unités de compte dépend directement des performances des marchés financiers et des choix de gestion effectués. La stratégie de gestion peut varier en fonction du profil de risque et des objectifs de l'investisseur. En gestion active, les gestionnaires de fonds prennent des décisions actives pour acheter et vendre des actifs afin de maximiser le rendement, en fonction des conditions du marché. Tandis qu'en gestion passive, la stratégie consiste à répliquer la performance d'un indice de marché en investissant dans les mêmes proportions que celles de l'indice, offrant ainsi une exposition diversifiée et souvent moins coûteuse.

Rendement Le profil de risque des unités de compte est généralement plus élevé que celui des fonds euros, mais il offre également un potentiel de rendement supérieur. Les performances passées ne garantissent pas les performances futures, et il est possible de subir des pertes en capital. Les épargnants doivent être conscients de leur tolérance au risque et des horizons de placement à long terme. Néanmoins, des garanties peuvent être ajoutées à ces contrats afin de fournir un minimum de protection au capital investi, par exemple les garanties planchers.

1.2.3 Contrat d'épargne hybride : Multisupport et Eurocroissance

Contrat multisupports A ces deux types de capitalisation s'ajoute une troisième famille de contrats : les contrats multisupports qui combinent le fonctionnement des fonds euros et des unités de comptes. Cette typologie de contrat permet à l'épargnant de diversifier son investissement en répartissant son capital entre des supports sécurisés (fonds en euros) et des supports dynamiques (unités de compte). Ainsi, une plus grande flexibilité est offerte à l'épargnant.

Contrats Euro-croissance Le contrat multisupport joue sur la combinaison de deux environnements d'investissement, tandis que le contrat Euro-croissance fait évoluer les garanties au cours de la durée de vie du contrat. Il combine une garantie partielle du capital avec un potentiel de rendement supérieur à long terme :

- Garantie partielle du capital : le capital est garanti, mais seulement à l'échéance du contrat, et non pas chaque année comme pour les contrats en fonds Euros.
- Potentiel de rendement supérieur : par une gestion plus flexible et une exposition aux marchés financiers.
- Effet de cliquet à long terme : La garantie du capital à l'échéance permet aux gestionnaires de fonds de prendre des positions plus risquées et potentiellement plus rémunératrices.

Les avantages fiscaux Les contrats d'assurance-vie sont le premier moyen d'épargne des Français par les avantages fiscaux dont ces contrats bénéficient. Les intérêts et plus-value accumulés dans un contrat d'assurance-vie peuvent être exonérés d'impôts sur le revenu jusqu'à un certain seuil ou dans certaines conditions (exonération possible huit ans après la souscription).[fin]

La fiscalité de l'assurance-vie influence directement la gestion des contrats et la planification patrimoniale. Tout d'abord, la fiscalité des retraits, ou rachats, joue un rôle crucial. En effet, avant 8 ans à compter de la souscription, le souscripteur peut choisir entre intégrer les intérêts et les plus-values à son impôt sur le revenu ou opter pour un prélèvement forfaitaire de 12,80% avec des prélèvements sociaux de 17,2%. Après 8 ans, les plus-values des versements effectués sont soumises à une imposition réduite de 7,5% avec des prélèvements sociaux de 17,2% après un abattement annuel de 4 600 € pour une personne seule et 9 200 € pour un couple. Les

rentes viagères issues de l'assurance-vie bénéficient également d'un abattement fiscal variable en fonction de l'âge du souscripteur, allant de 50% pour les souscripteurs entre 50 et 60 ans à 70% pour ceux de plus de 70 ans.

En cas de décès du souscripteur, les avantages fiscaux pour les bénéficiaires dépendent de l'âge du souscripteur et de la date des versements. Les versements effectués avant les 70 ans du souscripteur bénéficient d'un abattement de 152 500 € par bénéficiaire. Les sommes excédant cet abattement sont taxées à un taux de 20% jusqu'à 700 000 € puis à 31,25% au-delà. Les versements effectués après les 70 ans du souscripteur sont réintégrés dans l'actif successoral et soumis aux droits de succession, après un abattement global de 30 500 €, tout en conservant une exonération des intérêts et des plus-values.

Ces mécanismes fiscaux ont un impact important sur la modélisation des flux futurs, les provisions techniques et la gestion du passif pour les assureurs. L'âge du souscripteur, le moment des versements et le choix des bénéficiaires modifiables à tout moment sont autant de paramètres qui influencent la structure des produits d'assurance-vie proposés. L'assurance-vie devient alors un outil d'optimisation patrimoniale, notamment grâce à la possibilité de transmission hors succession. En intégrant ces éléments dans les modèles actuariels, les compagnies d'assurance peuvent améliorer leur gestion des risques et proposer des produits adaptés aux objectifs de leurs clients tout en tenant compte des avantages fiscaux.

1.3 Contexte réglementaire

La partie précédente a présenté la situation du secteur de l'assurance-vie en France. Cette section s'intéresse aux normes législatives imposées aux compagnies d'assurance pour répondre à leurs engagements, et plus particulièrement à la définition du capital de solvabilité requis.

1.3.1 Solvabilité I puis II

Serge Braudo dans son dictionnaire juridique [Bra] définit la solvabilité comme "la capacité d'une personne de disposer des moyens suffisants lui permettant de s'acquitter de ses dettes certaines, liquides et exigibles". Cette notion de solvabilité est essentielle dans le secteur de l'assurance en raison du cycle de production inversé. En effet, la compagnie d'assurance reçoit des primes avant de réaliser ou non des prestations. D'où la nécessité pour une compagnie d'assurance de se constituer un capital de solvabilité pour faire face à ses engagements futurs. L'assureur doit donc suivre en permanence l'évolution de son risque de solvabilité. Il doit alors identifier les sources possibles : tarification, dérive de sinistralité, fluctuations des marchés financiers...

En 1973, l'Union Européenne adopte la directive Solvabilité I afin de réglementer les pratiques et de mieux protéger les assurés. Cette directive reposait sur des principes simples et des règles fixes, essentiellement basées sur des exigences de capital minimum et des provisions techniques standardisées. Malgré sa simplicité et son efficacité initiale, la directive présentait plusieurs limitations significatives qui ont motivé la transition vers Solvabilité II. Tout d'abord, son manque de sensibilité au risque : les profils de risque spécifiques des compagnies d'assurance n'étaient pas pris en compte dans les exigences en capital. Cette approche uniforme pouvait mener à une surévaluation ou une sous-évaluation des besoins en capital. De plus, Solvabilité I offrait peu d'incitation à la gestion proactive des risques, car le cadre réglementaire ne récompensait pas les entreprises adoptant des pratiques sophistiquées de gestion des risques. Enfin, l'absence de flexibilité dans le cadre de Solvabilité I rendait difficile l'adaptation aux évolutions du marché et aux innovations dans le secteur de l'assurance, ce qui limitait la capacité des entreprises à répondre efficacement aux nouvelles menaces et opportunités.

Le 1er janvier 2016, la directive Solvabilité II entre en application, avec l'objectif d'aligner les exigences de capital sur les risques réels auxquels les assureurs sont exposés. Solvabilité II introduit une approche plus complexe et basée sur le risque. Elle incite à une meilleure gestion du risque et renforce la protection des assurés. Cependant, elle soulève de nouveaux enjeux pour les assureurs : la complexité et le coût de la méthode.

1.3.2 Les trois piliers de Solvabilité II

La directive se décompose en trois piliers :

1. Exigences quantitatives
2. Exigences qualitatives
3. Discipline de marché

Pilier 1 : Exigences quantitatives Le pilier 1 regroupe les exigences quantitatives, c'est-à-dire les règles de valorisation des actifs et des passifs, ainsi que les exigences de capital et leur mode de calcul. Les actifs et les passifs sont désormais présentés en valeur de marché. La directive impose deux exigences de capital :

— le minimum de capital requis (Minimum Capital Requirement - MCR)

— le capital de solvabilité requis (Solvency Capital Requirement - SCR)

Dans le cadre de ce mémoire, ce pilier est essentiel, car il définit les règles de calcul des indicateurs à optimiser. Le SCR peut être calculé au moyen d'une formule standard ou d'un modèle interne complet ou partiel. Dans le cadre du mémoire, le SCR sera évalué via la formule standard. [ACP]

Pilier 2 : Exigences qualitatives Le Pilier 2 de Solvabilité II regroupe les règles de gouvernance et de gestion des risques, ainsi que l'évaluation interne des risques et de la solvabilité à travers l'ORSA (Own Risk and Solvency Assessment). Ce pilier impose aux organismes d'assurance la mise en place d'un système de gouvernance efficace pour garantir une gestion saine et prudente de leurs activités. Ce système doit inclure au minimum deux dirigeants effectifs et quatre responsables de fonctions clés : fonction actuarielle, gestion des risques, audit interne et conformité. En complément du contrôle interne, le pilier 2 codifie le contrôle externe exercé par le superviseur européen, qui peut imposer une marge additionnelle de capital (capital add-on) en cas d'inadéquation entre la formule standard ou le modèle interne et le profil de risque de l'organisme, ou en cas de défaillance de la gouvernance ou de la gestion des risques. L'ORSA permet à l'organisme d'évaluer sa capacité à identifier, mesurer et gérer les risques susceptibles de modifier sa solvabilité ou sa situation financière. En tant qu'outil stratégique, l'ORSA doit être utilisé pour piloter l'activité en fonction des risques identifiés, renforçant ainsi la résilience et la stabilité des entreprises d'assurance.[ACP]

Pilier 3 : Discipline de marché Le Pilier 3 de Solvabilité II se concentre sur la communication d'informations au public et aux autorités de contrôle, visant à harmoniser les publications des organismes d'assurance au niveau européen. Ce pilier impose des exigences de divulgation rigoureuses et cohérentes, tant comptables que réglementaires. Les rapports, notamment le Rapport de Solvabilité et de Condition Financière (SFCR) pour le public et le Rapport Régulier au Superviseur (RSR) pour les régulateurs, doivent inclure des informations quantitatives et qualitatives sur la situation financière, la gouvernance, la gestion des risques et la solvabilité de l'entreprise. Ces informations doivent être publiées annuellement et, pour certaines, trimestriellement. En garantissant la cohérence et l'harmonisation des informations à l'échelle de l'Union européenne, le pilier trois vise à renforcer la transparence, à permettre des décisions informées par les parties prenantes et à promouvoir la stabilité et la résilience du secteur de l'assurance.

1.3.3 Focus sur le pilier 1 : Le bilan passif prudentiel

En vue de l'importance de ce pilier pour le sujet d'étude, cette sous-section détaille la composition du bilan prudentiel au passif.

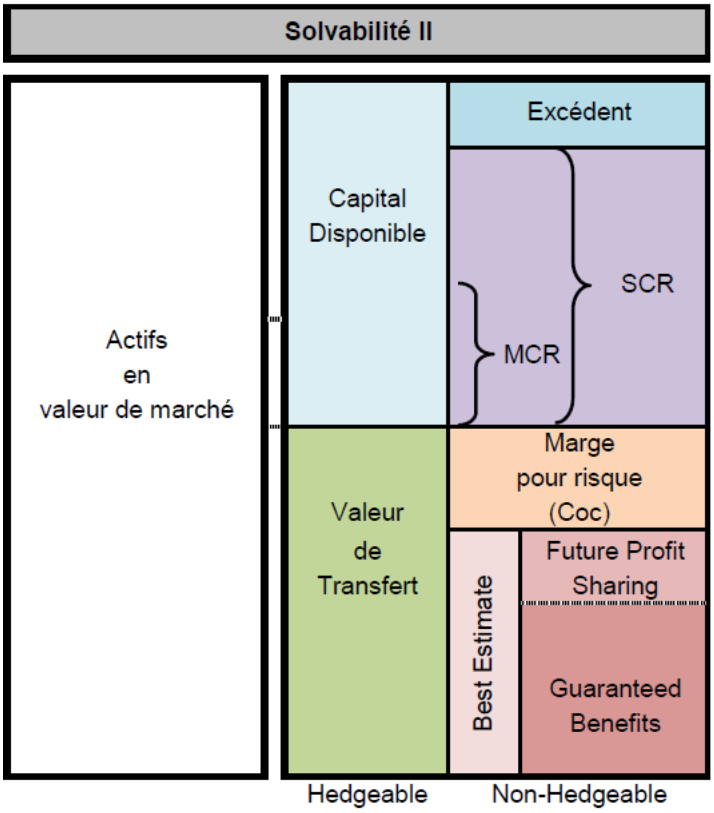

FIGURE 7 – Bilan prudentiel sous solvabilité II

Best estimate le Best Estimate se définit comme la valeur actuelle probable des flux de trésorerie futurs liés à l'exécution des contrats en portefeuille. L'actualisation se fait avec une courbe de taux sans risque fournie par l'EIOPA. Le Best Estimate correspond au montant que doit détenir un assureur pour être en mesure d'indemniser ses assurés « selon une vision moyenne ». Il doit prendre en compte toutes les options et garanties offertes par le contrat en anticipant le comportement probable des assurés, l'ensemble des versements discrétionnaires futurs (notamment la participation aux bénéfices), et ce, sur l'ensemble des contrats existants que l'assureur ne peut unilatéralement refuser.

$$\text{BE} = \sum_{i=0}^{\infty} \frac{\mathbb{E}[CF_i]}{(1+r_i)^i}$$

Avec :

- CF_i les cash-flows l'année i
- r_i le taux sans risque pour la maturité i

Cette courbe de taux sans risque est construite à partir des taux swaps « euro contre EURIBOR 6 mois ». Un ajustement de risque de crédit (CRA), compris entre 0,10 % et 0,35% , est appliqué aux taux observés jusqu'au Last Liquid Point (LLP), fixé à 20 ans.

$$\text{Risk Free Rate} = \text{Swap rate} - \text{Credit Risk Adjustment}$$

La courbe est alors construite par interpolation sur les maturités inférieures au Last Liquid Point puis extrapolée pour converger vers l'ultimate forward rate (UFR) à horizon 60 ans. Un ajustement lié à la volatilité est également appliqué à la courbe pour limiter l'impact de la volatilité des marchés financiers. [Eur][IA16]

En décembre 2023, le volatility adjustment est égale à 20 bps, et le credit risk adjustment s'élève à 10 bps. L'Ultimate forward rate, quant à lui, se fixe à 3,45%.

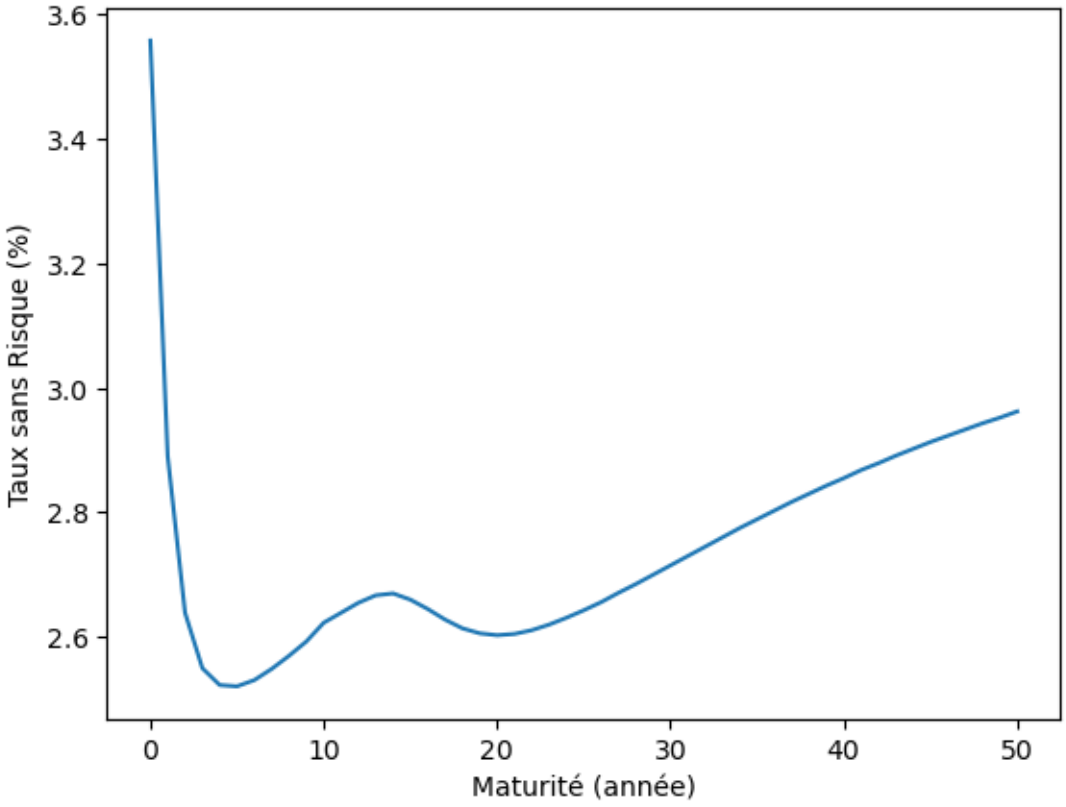

FIGURE 8 – Courbe Taux sans risque au 31/12/2023 avec VA

Cette courbe de taux témoigne d'un contexte économique complexe. L'inversion de la courbe de taux n'encourage pas les investissements à long terme.

Risk Margin La Risk Margin constitue la seconde composante des provisions techniques. Conçu pour garantir que les obligations d'assurance puissent être transférées à une autre entreprise en cas de défaillance de l'assureur initial. Le Risk Margin couvre les coûts additionnels que le repreneur devrait supporter pour détenir le capital requis afin de soutenir les engagements de l'assureur initial jusqu'à leur extinction. Le Risk margin est calculé en actualisant le coût du capital annuel généré par l'immobilisation du SCR sur la durée de vie résiduelle des engagements utilisés pour le calcul du Best Estimate.

$$\mathrm{RM} = \mathrm{CoC} \sum_{t \geq 0} \frac{\mathrm{SCR}(t)}{(1 + r_{t+1})^{t+1}}$$

Où :

- Coc représente le taux de coût de capital, fixé à 6 % par [Eur]
- SCR(t) est le capital de solvabilité requis après t années
- r_{t+1} est le taux d'intérêt sans risque de base pour l'échéance t+1 années

Les fonds propres Les fonds propres jouent un rôle crucial dans la stabilité financière et la résilience des compagnies d'assurance. Ils servent de coussin de sécurité pour absorber les pertes imprévues et garantir que l'assureur peut honorer ses engagements envers les assurés, même

en cas de période de stress. Les fonds propres se divisent en trois niveaux (Tier 1, Tier 2, Tier 3) selon leur qualité et leur capacité à absorber les pertes. Le pilier 1 impose que les assureurs détiennent un niveau de fonds propres suffisants pour couvrir le capital de solvabilité requis (SCR), qui reflète les risques sous-jacents des actifs et des passifs. Les fonds propres sont intégrés dans la valeur de l'actif net (NAV), c'est-à-dire à la différence entre la valeur économique de l'actif et la valeur des provisions techniques (Best estimate). La valeur économique de l'actif se décompose en deux ensembles :

— Les fonds propres en vision historique : les capitaux propres sociaux
— La Value in Force : valeur actuelle des profits futurs retirée de la marge de risque

1.3.4 SCR et MCR

Le SCR (solvency capital requirement) représente le niveau de capital que les assureurs doivent maintenir pour couvrir les risques auxquels ils sont exposés et pour garantir qu'ils disposent des fonds nécessaires pour absorber les pertes imprévues (fluctuations des marchés financiers, catastrophes naturelles...). Le SCR se définit de façon à assurer une probabilité de défaillance inférieure à 0,5 % à horizon d'un an ($\mathbb{P}(NAV \leq 0) \leq 0.005$). Autrement dit, le SCR est le niveau de capital pour que la ruine de l'organisme sur l'année en cours soit un évènement bicentenaire.

Méthodes d'évaluation du SCR Le SCR peut être évalué suivant plusieurs méthodes :

— Le modèle standard fourni par l'EIOPA
— Le modèle interne développé par l'organisme

Formule standard Le modèle standard fourni par l'EIOPA est conçu pour être appliqué par tous les assureurs de manière uniforme. Ce modèle décompose le risque en plusieurs modules et sous-modules. Le capital requis pour chaque source de risque est évalué pour chaque sous-module puis agrégé au sein d'un module avant d'être agrégé sur l'ensemble des modules. Les sous-modules correspondent à des chocs appliqués à la compagnie d'assurance. Ainsi, le SCR associé à un sous-module se définit comme la différence de NAV du scénario central et du scénario choqué.

$$\mathrm{SCR}_{\text{sous module}} = \Delta\ \mathrm{NAV} = \mathrm{NAV}_{\text{centrale}} - \mathrm{NAV}_{\text{choquée}}$$

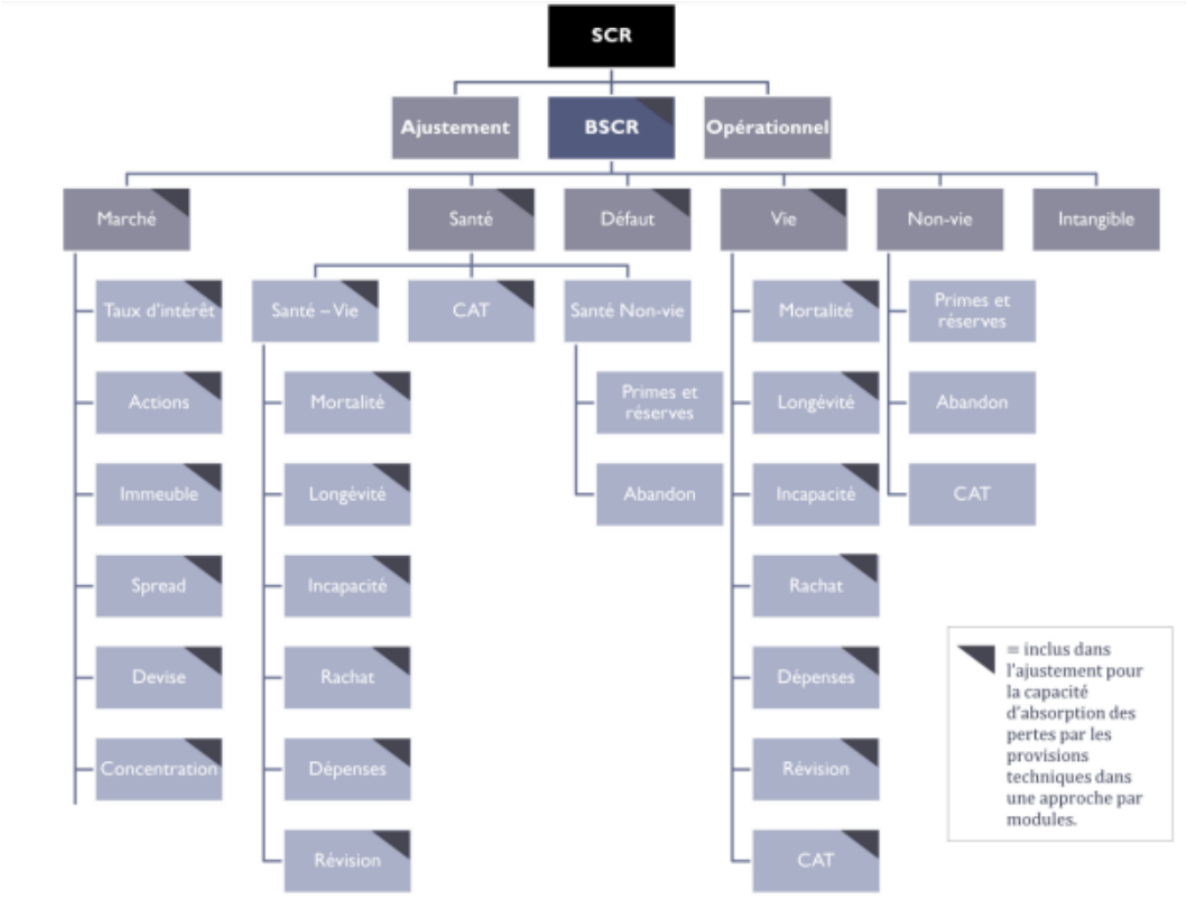

FIGURE 9 – Décomposition du SCR en modules et sous-modules

Le calcul du SCR se décompose en plusieurs étapes. Comme illustré sur la figure 9, le SCR est la combinaison de trois éléments :

- Basic SCR (BSCR) : constitue l'agrégat des SCRs des sous-modules
- Risque opérationnel (Op) : couvre les risques liés à des défaillances internes, des processus, des systèmes ou des personnes.
- Ajustements (Adj) : pour tenir compte de la capacité d'absorption des pertes par les provisions techniques et les impôts différés.

$$\text{SCR} = \text{BSCR} + \text{Op} - \text{Adj}$$

Le BSCR est lui-même l'agrégation de différents modules :

- **Risque de Marché** : représente la volatilité des marchés, entraînant des fluctuations de prix des instruments financiers. Ce risque impacte initialement les actifs de l'organisme, affectant leur valeur de marché, leur rendement et les opportunités de réinvestissement. Les passifs sont également influencés, de manière plus indirecte, par l'exercice des options et garanties des assurés, notamment dans le cadre des contrats en euros.
- **Risque de Défaillance** : le risque de contrepartie concerne la possibilité que les partenaires commerciaux d'une compagnie d'assurance ne respectent pas leurs obligations contractuelles. Ce risque peut émaner de diverses sources, telles que les transactions de produits dérivés, les opérations de réassurance, les prêts ou autres engagements financiers. Lorsqu'une compagnie d'assurance participe à ces transactions, elle s'expose au risque que la contrepartie fasse défaut, ce qui peut entraîner des pertes financières significatives.
- **Risque Vie** : Le module de souscription en assurance vie est conçu pour évaluer les

risques spécifiques associés aux contrats d'assurance-vie et aux engagements envers les assurés. Ce module prend en compte divers risques liés aux particularités des produits d'assurance-vie, notamment la mortalité, la longévité, la durée d'incapacité ou de morbidité, ainsi que les frais et les rachats, qui sont tous des éléments fondamentaux pour la tarification.

— **Risque Non-vie** : Le module de souscription en Non-Vie évalue les risques liés à l'activité de l'assurance non-vie, provenant directement des sinistres couverts ou d'une mauvaise gestion de l'activité. Ce risque inclut également l'incertitude concernant le comportement de l'assuré, comme la reconduction des contrats, ainsi que le risque de catastrophe (CAT), qui couvre la survenance de catastrophes naturelles (tsunami, cyclone, tempête, etc.) ou d'origine humaine (telle qu'une attaque terroriste).

— **Risque de Santé** : Le module de souscription en santé représente des risques similaires à ceux considérés en vie et en non-vie, mais spécifiquement pour les lignes de business liées à la santé. Une distinction est faite entre la santé SLT (Similar to Life Techniques), qui inclut les lignes de business pouvant être traitées via des méthodes similaires à celles utilisées pour le SCR Vie, et la santé non-SLT (Non Similar to Life Techniques), assimilable à la non-vie (comme les frais médicaux, l'indemnisation des travailleurs, la protection du revenu, etc.).

— **Risques Intangibles** : se rapportent aux actifs immatériels de l'assureur, tels que les brevets, les licences et autres droits de propriété intellectuelle, qui peuvent perdre de la valeur ou devenir inutilisables, impactant ainsi la solvabilité de l'assureur.

Pour chaque module, les SCRs s'agrègent ensuite en suivant la formule suivante :

$$\text{BSCR} = \sqrt{\sum_{i,j} \text{SCR}_i \ \text{Corr}\ _{ij} \text{SCR}_j} + \text{SCR}_{\text{intangibles}}$$

	SCR Marché	SCR Défaut	SCR Vie	SCR Santé	SCR Non-vie
SCR Marché	1	0.25	0.25	0.25	0.25
SCR Défaut	0.25	1	0.25	0.25	0.5
SCR Vie	0.25	0.25	1	0.25	0
SCR Santé	0.25	0.25	0.25	1	0
SCR Non-vie	0.25	0.5	0	0	1

TABLE 4 – Matrice corrélation pour l'agrégation des modules de SCR

La formule standard pour le calcul du SCR sous Solvabilité II présente des avantages notables, notamment sa simplicité, son uniformité, sa transparence, et son faible coût d'implémentation, ce qui facilite la conformité réglementaire et permet des comparaisons aisées entre assureurs. Cependant, elle comporte aussi des inconvénients majeurs : son manque de personnalisation peut entraîner une sur- ou sous-capitalisation en raison de paramètres génériques qui ne reflètent pas fidèlement les profils de risque spécifiques.

Modèle interne Pour améliorer l'adéquation entre l'évaluation du SCR et leur profil de risque, certaines compagnies d'assurance mettent en place des modèles internes. L'utilisation de ces modèles est soumise à une autorité de contrôle, telle que l'ACPR en France, qui doit approuver leur utilisation. Les modèles internes offrent une flexibilité accrue et permettent de refléter plus précisément les spécificités de chaque assureur. Deux principales approches peuvent être adoptées dans le développement des modèles internes :

— **Approche modulaire :** Cette approche est inspirée de la formule standard. Un exemple courant est l'utilisation des Undertaking Specific Parameters (USP), qui sont des paramètres propres à l'entité. Les USP permettent une personnalisation partielle en ajustant des variables spécifiques tout en utilisant le cadre général de la formule standard. Cela permet une meilleure représentation des risques spécifiques tout en conservant une structure standardisée.
— **Approche intégrée :** Contrairement à l'approche modulaire, l'approche intégrée considère l'ensemble des risques de manière globale plutôt que de les traiter individuellement. Elle implique une modélisation holistique des risques, prenant en compte les interactions et les corrélations entre différents types de risques (marché, crédit, opérationnel, etc.). Cette approche permet une vision plus cohérente et complète du profil de risque de l'assureur, favorisant une gestion des risques plus efficace et plus réactive.

La mise en place d'un modèle interne nécessite des investissements significatifs en termes de ressources humaines et technologiques. Le processus de validation est rigoureux, impliquant des tests de stress et des analyses de sensibilité pour s'assurer de la robustesse et de la fiabilité du modèle. Une fois approuvé, le modèle interne doit être utilisé de manière continue et faire l'objet de revues régulières pour garantir sa pertinence et son efficacité dans l'évaluation des risques et la détermination du SCR. Le recours à des modèles internes permet aux assureurs d'adopter une approche de gestion des risques plus sophistiquée, alignée sur leur profil de risque unique et leurs stratégies d'entreprise.

Détermination du MCR Le minimal capital requirement (MCR) est le seuil minimal de fonds propres à détenir en dessous duquel l'autorité de contrôle intervient pour redresser ou transférer les engagements de l'assureur. Les articles 258 à 251 du règlement délégué [Eur] définissent la méthode de calcul. Dans le cadre de l'assurance-vie, le MCR s'exprime :

$$\mathrm{MCR} = \max(\mathrm{MCR}_{\text{combined}}, \mathrm{AMCR})$$
$$\mathrm{MCR}_{\text{combined}} = \min(\max(\mathrm{MCR}_{(\text{linear,l})}, 0.25\ \mathrm{SCR}), 0.45\ \mathrm{SCR})$$
$$\mathrm{MCR}_{(\text{linear,l})} = 0.037\ \mathrm{TP}_{(\text{life,1})} - 0.052\ \mathrm{TP}_{(\text{life,2})} + 0.007\ \mathrm{TP}_{(\text{life,3})} + 0.021\ \mathrm{TP}_{(\text{life,4})} + 0.0007\ \mathrm{CAR}$$

où :
— AMCR est le seuil plancher absolu fixé ici à 2 M€
— $\mathrm{TP}_{(\text{life,i})}$: des provisions techniques (Article 251 [Eur]
— CAR :le montant total du capital sous risque

1.3.5 Focus SCR Marché

Dans la formule standard, le risque de marché se décompose en 6 modules :
— risque de taux d'intérêt
— risque actions
— risque immobilier
— risque de spread
— risque de devise
— risque de concentration

Les différents risques s'agrègent suivant la matrice de corrélation 5

Le paramètre A est égal à 0 si le scénario taux choqué à la hausse est celui sélectionné pour le calcul du SCR de taux. Sinon, A est égale à 0,5.

	Taux intérêt	Actions	Actifs immobiliers	Spread	Concentration	Devise
Taux intérêt	1	A	A	A	0	0.25
Actions	A	1	0.75	0.75	0	0.25
Actifs immobiliers	A	0.75	1	0.5	0	0.25
Spread	A	0.75	0.5	1	0	0.25
Concentration	0	0	0	0	1	0
Devise	0.25	0.25	0.25	0.25	0	1

TABLE 5 – Matrice corrélation pour l'agrégation des sous-modules du SCR marché

Risque de taux d'intérêt Ce risque représente la sensibilité des actifs et des passifs de l'assureur aux variations des taux d'intérêt. Une hausse ou une baisse des taux peut affecter la valeur actuelle des flux de trésorerie futurs des obligations, des prêts, ainsi que la valeur de marché des instruments financiers. Ce sous-module mesure l'impact potentiel de ces variations sur la solvabilité de l'assureur. Des coefficients de majorations et de minorations sont donc appliqués à la courbe de taux sans risque. Les chocs sont appliqués aux maturités non explicitement édictées, sont estimés par interpolation linéaire. Le règlement précise que l'écart de taux lors du choc à la hausse doit être d'au moins 1%, alors que lors du choc à la baisse, les taux négatifs ne sont pas affectés. Le SCR retenu est celui du scénario le plus défavorable.

Maturité (en années)	**Hausse**	**Baisse**
1	70%	75%
2	70%	65%
3	64%	56%
4	59%	50%
5	55%	46%
6	52%	42%
7	49%	39%
8	47%	36%
9	44%	33%
10	42%	31%
11	39%	30%
12	37%	29%
13	35%	28%
14	34%	28%
15	33%	27%
16	31%	28%
17	30%	28%
18	29%	28%
19	27%	29%
20	26%	29%
90	20%	20%

TABLE 6 – Tableau des maturités et des pourcentages de hausse et de baisse

Ces trois courbes serviront d'input au générateur de scénario économique pour l'estimation des NAVs.

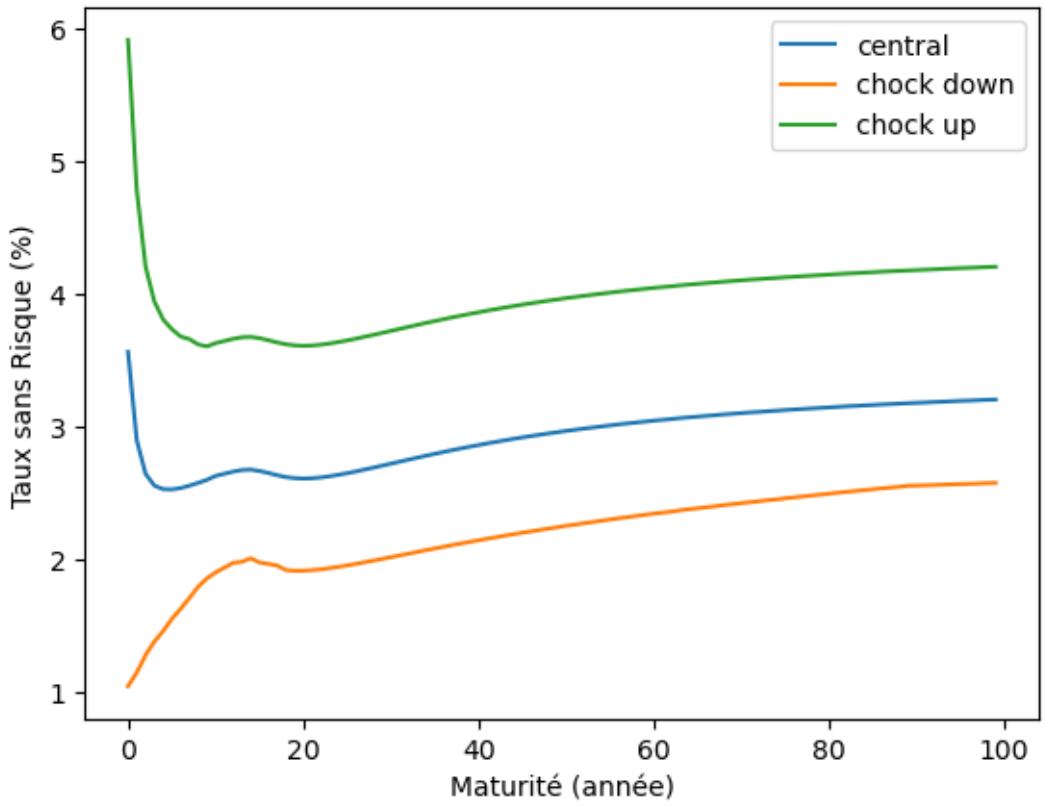

FIGURE 10 – Courbe de taux centrale et choquées au 31/12/2023

Risque actions Ce risque est lié aux fluctuations des prix des actions détenues par l'assureur. Ce sous-module évalue la sensibilité du portefeuille d'actions aux variations des marchés boursiers. Une chute des prix des actions peut entraîner une diminution significative de la valeur des actifs de l'assureur, affectant ainsi sa solvabilité. Un coefficient de minoration est appliqué à la valeur de marché des actions et autres actifs liés au marché des actions. La valeur de ce coefficient dépend de la nature de l'action :

- Les actions type 1, cotées sur un marché régulé dans un pays de l'OCDE ou de l'EEE : une décote de -39% est appliquée.
- Les actions type 2, non cotées ou cotées dans un pays hors de l'OCDE ou de l'EEE : une décote de -49% est appliquée.

Le choc est de 22% pour les placements en actions dans des entreprises liées et de nature stratégique. Pour tenir compte des effets cycliques du marché, un ajustement symétrique (AS), compris entre -10% et 10%, est appliqué.

$$AS = \frac{1}{2}(\frac{CI - AI}{CI} - 8\%)$$

où :

- CI le montant de l'indice actions au moment du calcul
- AI la moyenne pondérée des niveaux quotidiens de l'indice sur les trois dernières années

A la date du 31/12/2023, l'ajustement symétrique est proche de 1,5%.

Les actions de chaque type sont choquées séparément, puis le montant en capital requis est agrégé suivant la formule :

$$\text{SCR}_{\text{equity}} = \sqrt{\text{SCR}_{\text{type 1}}^2 + 1.5\ \text{SCR}_{\text{type 1}}\ \text{SCR}_{\text{type 2}} + \text{SCR}_{\text{type 2}}^2}$$

Risque immobilier Ce risque concerne les variations des valeurs des actifs immobiliers détenus par l'assureur. Les fluctuations des prix de l'immobilier peuvent résulter de changements dans les conditions économiques, les taux d'intérêt, et les dynamiques du marché immobilier. Ce sous-module mesure l'impact potentiel de ces variations sur le bilan de l'assureur. Comme pour le risque actions, le capital requis est évalué en diminuant la valeur nette de l'actif de 25%.

Risque de spread Ce risque est lié aux variations des spreads de crédit, c'est-à-dire les différences de rendement entre les obligations d'entreprises (ou autres titres de crédit) et les obligations d'État sans risque. Cet écart rémunère le risque pris par le détenteur d'une obligation potentiellement moins liquide et avec un risque de défaut plus grand. Une augmentation des spreads de crédit peut indiquer une perception accrue du risque de défaut, entraînant une baisse de la valeur de ces obligations. Ce sous-module évalue l'impact de ces variations sur les actifs de l'assureur. La réglementation définit un tableau pour appliquer des chocs à la valeur de marché des obligations en fonction de leur duration modifiée et de leur notation. 7

Qualité de crédit		**0**		**1**		**2**		**3**		**4**		**5 et 6**	
Duration (dur_i)	stress$_i$	a_i	b_i	a_i	b_i	a_i	b_i	a_i	b_i	a_i	b_i	a_i	b_i
Jusqu'à 5 ans	$b_i \cdot dur_i$	—	0,9%	—	1,1%	—	1,4%	—	2,5%	—	4,5%	—	7,5%
Supérieure à 5 et inférieure ou égale à 10 ans	$a_i + b_i \cdot (dur_i - 5)$	4,5%	0,5%	5,5%	0,6%	7,0%	0,7%	12,5%	1,5%	22,5%	2,5%	37,5%	4,2%
Supérieure à 10 et inférieure ou égale à 15 ans	$a_i + b_i \cdot (dur_i - 10)$	7,0%	0,5%	8,5%	0,5%	10,5%	0,5%	20,0%	1,0%	35,0%	1,8%	58,5%	0,5%
Supérieure à 15 et inférieure ou égale à 20 ans	$a_i + b_i \cdot (dur_i - 15)$	9,5%	0,5%	11,0%	0,5%	13,0%	0,5%	25,0%	1,0%	44,0%	0,5%	61,0%	0,5%
Plus de 20 ans	$\min[a_i + b_i \cdot (dur_i - 20); 1]$	12,0%	0,5%	13,5%	0,5%	15,5%	0,5%	30,0%	0,5%	46,6%	0,5%	63,5%	0,5%

TABLE 7 – Coefficient de choc à appliquer à la valeur de marché des obligations en fonction de la maturité et la sensibilité pour le SCR de spread

Les obligations ou prêts pour lesquels une évaluation de crédit par un organisme externe d'évaluation du crédit n'est pas disponible, et qui n'ont pas fourni de sûreté conforme aux critères énoncés à l'article 214 du règlement délégué, se voient attribuer un facteur de stress en fonction de leur sensibilité et duration conformément au tableau 8

duration (dur_i)	**stress**$_i$
Jusqu'à 5 ans	$3\% \cdot dur_i$
Supérieure à 5 et inférieure ou égale à 10 ans	$15 + 1,7\% \cdot (dur_i - 5)$
Supérieure à 10 et inférieure ou égale à 20 ans	$23,5\% + 1,2\% \cdot (dur_i - 10)$
Plus de 20 ans	$\min(35, 5\% + 0,5\% \cdot (dur_i - 20); 1)$

TABLE 8 – Coefficients de choc à appliquer à la valeur de marché des obligations non notées

Risque de devise Ce risque concerne les fluctuations des taux de change, affectant les actifs et passifs libellés en devises étrangères. Les variations des taux de change peuvent entraîner des gains ou des pertes sur les positions en devises, affectant ainsi la solvabilité de l'assureur. Ce sous-module mesure l'impact potentiel de ces fluctuations sur le bilan de l'assureur. L'exigence en capital est égale à la somme de l'exigence en capital de chaque monnaie. Cette exigence en capital se définit comme le maximum du capital requis en cas de choc à la hausse de 25% et capital requis en cas de choc à la baisse de 25% est appliqué sur le taux de change.

Risque de concentration Ce risque se rapporte à la perte significative pouvant résulter d'une forte concentration des investissements dans certains actifs, secteurs, émetteurs ou géographies. Une exposition excessive à un seul actif ou émetteur peut accroître la vulnérabilité de l'assureur aux chocs spécifiques à cet actif ou émetteur. Ce sous-module évalue l'impact de la concentration sur un émetteur sur la solvabilité de l'assureur. Un choc g_i (9) est appliqué sur l'exposition en excès d'actifs issus d'un même émetteur.

Moyenne pondérée des échelons de qualité de l'exposition sur signature unique i	0	1	2	3	4	5	6
Facteur de risque g_i	12 %	12 %	21 %	27 %	73 %	73 %	73 %

TABLE 9 – Facteur de risque selon la moyenne pondérée des échelons de qualité

L'exposition en excès se définit à partir d'un seuil CT_i fixé selon l'échelon de qualité moyen des actifs visés, de même que le choc à appliquer sur cet excédent 9 :

$$XS_i = \max(0; E_i - CT_i \times \text{Assets})$$

Avec :
— XS_i, l'exposition en excès
— E_i, l'exposition à la signature unique i
— CT_i, le seuil relatif d'exposition en excès
— Assets, base de calcul du sous-module

Moyenne pondérée des échelons de qualité de l'exposition sur signature unique i	0	1	2	3	4	5	6
Seuil relatif d'exposition en excès CT_i	3%	3%	3%	1.5%	1.5%	1.5%	1.5%

TABLE 10 – Seuil relatif d'exposition en excès en fonction de la notation pondérée

Les actifs dont l'émetteur est la BCE ou un État avec une notation de 0 ou 1 ne sont pas sensibles à ce choc. Le choc est également réduit pour les actifs issus d'État noté 2 ou plus.

$$\text{SCR}_{\text{concentration}} = \sqrt{\sum_i (XS_i \times g_i)}^2$$

Hypothèses de calcul du SCR Dans le cadre du mémoire, le SCR sera calculé en utilisant uniquement les risques suivants de la composante Marché :

- Risque action
- Risque de taux
- Risque immobilier
- Risque de spread

Le risque de change n'est pas pris en compte, car le portefeuille étudié ne contient que des titres en euros. Ensuite, faute de données suffisamment précises sur la détention des titres, le risque de concentration n'a pas été modélisé.

2 Modèle et hypothèses ALM

Piermay, Mathoulin et Cohen définissent le modèle ALM (asset liability management) de la manière suivante :

> La gestion actif-passif consiste d'une part à analyser la couverture des engagements d'un assureur ou d'un investisseur institutionnel par les actifs dans une perspective de déroulement dans le temps. Elle regroupe d'autre part l'ensemble des moyens d'action visant à piloter le bilan de l'institution. L'absence de gestion actif-passif aussi bien au sens de l'analyse que de l'action est pour beaucoup dans les difficultés rencontrées par les assureurs et les fonds de pensions au cours des quinze dernières années dans différents pays : couverture d'engagements certains par espoir de plus-values, prolongation de la tendance passée, absence d'examen de scénarios d'évaluation des actifs et des passifs.

La modélisation ALM (Asset-Liability Management) implique de modéliser simultanément les actifs, les passifs et leurs interactions au sein d'une compagnie d'assurance-vie. Cette modélisation intègre les décisions stratégiques dynamiques de l'entreprise, telles que la participation aux bénéfices, l'allocation des actifs, la réalisation de plus ou moins-values latentes et la constitution de provisions.

2.1 Présentation générale de Prophet

Utilisé par plus de 10 000 professionnels de l'assurance dans plus de 1 000 sites répartis dans plus de 80 pays, Prophet aide les entreprises à satisfaire leurs obligations de reporting, à améliorer la gestion des risques et à développer plus rapidement des produits plus rentables. Prophet est une solution actuarielle d'entreprise développée par FIS, largement reconnue et utilisée dans l'industrie des assurances et des services financiers.

Prophet propose une gamme complète de fonctionnalités pour les calculs actuariels. Parmi ses fonctionnalités, on retrouve la tarification et la conception de produits, la budgétisation et la planification des affaires, les évaluations statutaires traditionnelles et les rapports financiers conformes aux normes IFRS, US GAAP et autres. De plus, Prophet permet la modélisation actif-passif dans les cadres suivants :

- les exercices de partage des bénéfices
- la prise de décision stratégique
- les projections stochastiques pour les calculs de valeur intrinsèque
- les tests de stress et de scénarios pour les régimes de capital basés sur le risque, comme Solvabilité II

Dans le cadre du mémoire, ce module sera utilisé afin d'optimiser les indicateurs réglementaires.

L'application Prophet Professional gère et stocke le code écrit par les actuaires pour définir des calculs notamment nécessaire aux projections des actifs et des passifs. Prophet se compose de deux principaux composants :

- L'interface utilisateur permet le développement des modèles actuariels et le lancement de simulations.
- Le moteur de calcul convertit le langage Prophet en code machine exécutable. Ce moteur de calcul offre également des fonctions d'entrée utilisateur et de stockage des résultats, ainsi que le transfert des valeurs entre les différents composants du modèle.

Cette architecture permet une exécution efficace et évolutive des modèles actuariels, essentielle pour répondre aux besoins complexes des entreprises d'assurance.

En conclusion, Prophet est un outil puissant et polyvalent, essentiel pour les actuaires et les gestionnaires de risques dans l'industrie des assurances et des services financiers. Ses capacités de modélisation financière intégrée et de gestion des données permettent aux entreprises de rationaliser leurs processus actuariels et de prendre des décisions informées pour une gestion optimale des risques et une rentabilité accrue. [Pro]

Dans le cadre du mémoire, un modèle ALM développé en interne par Deloitte sur Prophet est utilisé.

2.2 Évaluation des actifs

Selon l'article 75 de la Directive Solvabilité II, les actifs doivent être évalués à leur valeur de marché. Cette valeur correspond au montant pour lequel les actifs pourraient être échangés dans le cadre d'une transaction réalisée dans des conditions de concurrence normales entre des parties informées et consentantes. Cela implique une évaluation basée sur les données disponibles sur les marchés financiers. Si ces données ne sont pas accessibles ou inexistantes, il est nécessaire de reconstituer les flux financiers de l'actif en question à partir de différents actifs similaires dont les prix sont connus. Cette approche, appelée évaluation en valeur de marché ou "market consistent", repose sur le principe de l'absence d'opportunités d'arbitrage, c'est-à-dire la possibilité de réaliser un gain certain en exploitant un écart de prix entre les différents marchés ou instruments financiers.

Dans le cadre de ce mémoire, les actifs sont représentés par des "model points", des objets regroupant des actifs présentant des caractéristiques similaires. Les "model points" se scindent entre trois catégories : les actions, les obligations et les biens immobiliers. Les actions et les biens immobiliers se caractérisent par leur valeur comptable et leur valeur de marché. Pour les obligations, il faut également ajouter les maturités, les coupons, les montants nominaux, les notations et le type d'émetteur (gouvernemental ou corporatif). Dans le modèle étudié, les obligations sont à taux fixe.

Dans le cadre de ce mémoire, l'allocation des actifs, est ventilée de la manière suivante pour optimiser certains indicateurs financiers, comme le SCR, le taux de liquidité :

- Action
- Obligation
- Immobilier
- Cash

2.3 Évaluation des passifs

Conformément à l'article 75 de la Directive Solvabilité II, les passifs doivent être évalués selon les mêmes conditions que les actifs, c'est-à-dire à leur valeur de marché. Pour le calcul du Best Estimate, il est essentiel d'utiliser des hypothèses statistiques et actuarielles adéquates et pertinentes. Lorsque les contrats d'assurance comprennent des options et des garanties, une approche stochastique est nécessaire pour refléter le caractère incertain des risques. En effet, les fluctuations de l'environnement économique influencent directement le comportement des assurés, notamment au travers des rachats et des souscriptions qui dépendent des performances du contrat (participation au bénéfice).

Le calcul du coût des options et garanties ne peut souvent pas se faire via des formules fermées ou des approximations basées sur un portefeuille d'actifs. Pour cette raison, une méthode de simulation de type Monte Carlo est recommandée pour sa simplicité. Cette approche permet de modéliser de manière plus précise et réaliste les différentes trajectoires possibles de l'évolution des paramètres économiques et financiers et leur impact sur les passifs de l'assurance.

Les données relatives au portefeuille des assurés sont également structurées sous forme de "model points", ce qui permet de regrouper les contrats d'assurance ayant des caractéristiques similaires. Les principales caractéristiques utilisées pour constituer ces points de modélisation sont :

- L'âge des assurés : Permet d'évaluer les risques en fonction de la tranche d'âge
- Le sexe des assurés : Influence les hypothèses de mortalité et de morbidité
- La date de souscription du contrat : Utile pour déterminer la durée du contrat et les engagements pris
- Le nombre de contrats : Indique la volumétrie et la concentration des risques
- Le montant moyen des provisions mathématiques au début de la projection : Reflète l'engagement financier initial
- Le taux minimum garanti : Impacte la valorisation des passifs en garantissant un rendement minimal
- Les primes programmées : Indiquent les contributions futures des assurés

Dans le cadre de ce mémoire, un passif théorique a été créé afin de représenter fidèlement la composition d'un portefeuille type d'un assureur vie en France. Ce passif est structuré en quatre "model points", chacun représentant différentes tranches d'âge allant de 40 à 70 ans, avec une moyenne d'âge située autour de 55 ans, reflétant ainsi l'âge moyen des détenteurs de contrats d'assurance-vie en France.

Chaque "model point" est conçu pour représenter un encours moyen de 44 000 € par contrat, avec un total de 1 000 contrats par point de modélisation, ce qui assure une cohérence dans la représentation des différentes tranches d'âge. Les taux d'intérêt minimum garantis varient en fonction de l'âge : 0.5 % pour les assurés de 40 ans, 1 % pour ceux de 50 ans, et 2 % pour les assurés de 60 et 70 ans.

Les provisions pour participation aux bénéfices et la réserve de capitalisation sont respectivement estimées à 6,0 millions d'euros et 0,9 million d'euros, représentant 3.5% et 0.5% des provisions mathématiques totales. Pour des raisons de simplification, d'autres provisions comptables, telles que la provision globale de gestion, sont considérées comme négligeables dans cette modélisation.

2.4 Générateur de scénarios économiques (GSE)

En assurance vie, afin d'estimer au mieux les engagements de l'assureur et le comportement de l'assuré, une approche stochastique est nécessaire. Pour simuler divers environnements économiques et financiers. Un GSE produit des scénarios cohérents de variables économiques clés comme :

- les taux d'intérêts
- les taux de change
- les rendements des actions
- les taux d'inflation

Présentation XSG XSG est une solution logicielle de génération de scénarios économiques développés par le cabinet Deloitte. Elle est conçue pour répondre aux besoins actuels et évolutifs en matière de modélisation de scénarios Monte Carlo pour les compagnies d'assurance et autres institutions financières. XSG propose des bibliothèques de modèles financiers et statistiques pour soutenir la modélisation stochastique en mode risque neutre et en mode monde réel dans une solution intégrée. XSG offre différentes typologies de modèles :

- Modèles en risque neutre : Utilisés pour la valorisation des options et garanties dans les passifs d'assurance. Ces modèles permettent de générer des scénarios proches des prix de marché des dérivés, avec une faible erreur d'échantillonnage et un support pour des rendements initiaux négatifs par la méthode de Monte Carlo.
- Modèles en monde réel : Calibrés sur des données historiques pour produire des projections réalistes, ces modèles peuvent être ajustés avec des hypothèses spécifiques.
- Modèles de risque à une étape : Génèrent des scénarios pour des conditions futures à partir de distributions statistiques sur une période d'un an.

Dans le cadre de ce mémoire, les modèles en risque neutre de XSG seront utilisés. Dans l'univers risque neutre, les investisseurs sont neutres au risque : ils n'exigent pas de prime de risque et s'attendent à recevoir un rendement égal à celui des taux sans risque.

2.5 Hypothèses sur les assurés et les assureurs

Pour obtenir une projection complète, il est nécessaire de modéliser non seulement les données sur les actifs, les passifs et l'environnement économique, mais aussi les comportements des assurés et des assureurs.

Assurés

- **Mortalité :** En assurance-vie, la sinistralité des assurés correspond à leur mortalité. Pour la modélisation de cette dernière, aucune table réglementaire n'est imposée. Ainsi, les assureurs peuvent utiliser des tables basées sur leur propre expérience de portefeuille. Cependant, l'organisme doit justifier de la bonne adéquation de la table de mortalité avec son risque. Dans le cadre du mémoire, la table réglementaire TGF05 sera utilisée.
- **Rachats et Arbitrages :** Les comportements des assurés en matière de rachats (partiels ou totaux) et d'arbitrages (entre fonds en euros et unités de compte) doivent être modélisés en tenant compte de phénomènes structurels et conjoncturels.
  - **Rachats structurels :** Les avantages fiscaux des contrats d'assurance-vie (notamment la défiscalisation après huit ans) influencent le rachat. Pour estimer ce rachat structurel, une méthode consiste à observer une moyenne mobile des rachats en fonction de l'ancienneté des contrats.

— **Rachats et arbitrages conjoncturels :** Ces phénomènes sont causés entre autre par la différence entre le taux servi et le taux concurrentiel. En effet, l'insatisfaction des assurés est le déclencheur de ces rachats et arbitrages.

Assureur Le modèle doit également inclure la modélisation des montants initiaux de réserves ainsi que les futures décisions de gestion de l'assureur. Cela comprend la gestion de la participation aux bénéfices (taux cible), la stratégie d'allocation des actifs et la stratégie de réinvestissement. Ces décisions doivent être cohérentes avec la stratégie actuelle de l'assureur ou avec les évolutions prévues.

— **Participation aux bénéfices :** La stratégie de participation aux bénéfices, notamment le taux servi, doit être modélisée. Ce taux cible détermine la part des profits financiers (au-delà de 85%) reversée aux assurés, impactant directement leur comportement. Ce taux est calculé comme un pourcentage du taux moyen d'emprunt d'État (TME) pondéré par un coefficient de lissage pour éviter des variations trop brusques d'une année à l'autre. La politique de reprise et de dotation de la provision pour participation aux bénéfices doit également être définie.

— **Allocation d'actifs :** La composition du portefeuille selon le type d'actif (obligations, actions, immobilier, cash) doit être définie. Dans le modèle utilisé, une stratégie fixe est utilisée, définie via une table Prophet où l'utilisateur indique le pourcentage d'obligations, d'actions, d'immobilier et de cash à détenir après réinvestissements pour chaque période.

— **Stratégie de réinvestissement :** Elle décide du processus à suivre pour suivre la stratégie d'allocation d'actifs. Elle permet de définir l'ordre par famille d'actifs dans lequel le réinvestissement doit être effectué et l'instant où la stratégie s'applique.

2.6 Fonctionnement général du modèle

Ainsi, le modèle se compose de deux briques, le GSE et le modèle ALM. Les scénarios construits par le GSE servent de paramètres au modèle ALM. Le modèle, en projetant le bilan de l'assureur, permet de calculer le SCR. (figure 11)

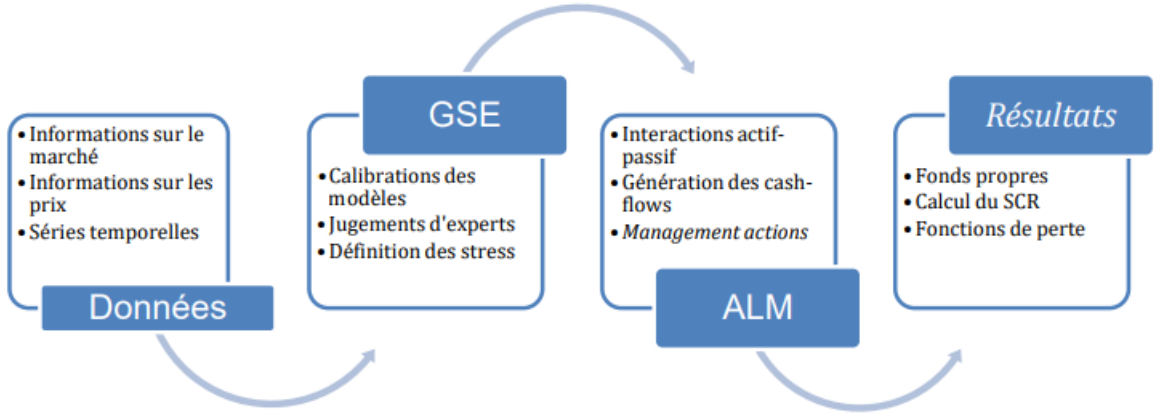

FIGURE 11 – Processus de calcul du SCR par le biais du modèle ALM

Ensuite, pour mieux comprendre le fonctionnement de Prophet, il faut étudier la chronologie des opérations qui s'effectue sur un exercice comptable d'une durée de 12 mois. Les différentes opérations sont détaillées sur la figure 12.

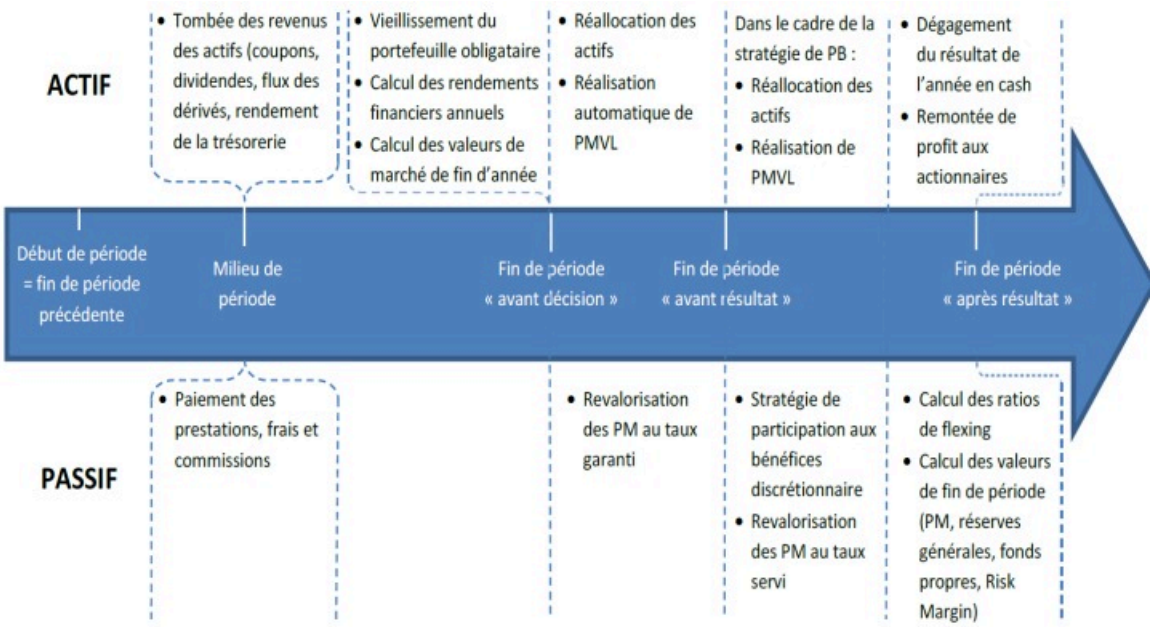

FIGURE 12 – Chronologie des opérations lors d'une projection

2.6.1 Fonctionnement de l'algorithme de réinvestissement

Le processus de réinvestissement suit l'algorithme décrit figure 13. En cas d'échec d'atteinte de l'allocation cible, le modèle réalise jusqu'à un maximum de 25 essais, au cours desquels il ajuste successivement chacune des N classes d'actifs pour atteindre l'allocation cible. Pour chaque classe d'actifs, il vérifie si la proportion détenue est conforme à la stratégie d'investissement définie. Si ce n'est pas le cas, il calcule le montant d'actifs à acheter ou vendre pour rééquilibrer l'allocation en fonction des actifs disponibles. Les actifs disponibles permettent de modéliser les actifs que l'assureur a la possibilité d'acheter sur les marchés. Ce processus se répète pour chaque classe d'actifs, en recalculant à chaque fois la proportion détenue après ajustement, jusqu'à ce que l'allocation cible soit atteinte, en traitant la trésorerie en dernier.

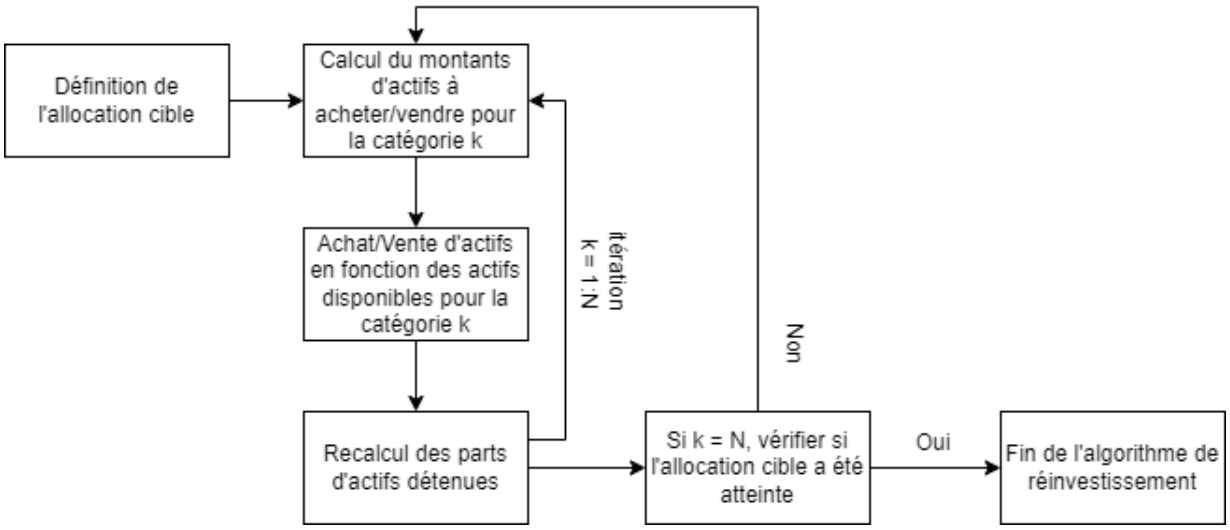

FIGURE 13 – Algorithme de réinvestissement

2.6.2 Méthode de flexing

Des méthodes de flexing sont utilisées pour réaliser cette étude. Le flexing s'appuie sur une projection en deux étapes : d'abord, les passifs sont projetés de façon déterministe en utilisant des hypothèses relatives à la mortalité, aux rachats structurels, aux frais et aux chargements des contrats. Seul le TMG (Taux Minimum Garanti) est pris en compte pour la revalorisation en l'absence de données sur les rendements financiers. Ainsi, une dynamique déterministe du comportement des passifs est estimée. Ensuite, pendant la projection stochastique, les flux de trésorerie déterministe vont être modifiés pour prendre en compte les interactions avec les actifs, les décisions de gestion et l'environnement économique. Cette phase permet de déterminer le montant de participation aux bénéfices reversés aux assurés et donc le comportement de ces derniers via les rachats conjoncturels.

Le flexing est une méthode intéressante, car elle simplifie les calculs et diminue le temps de projection. Mais elle perd en précision par rapport aux modélisations intégrées. En effet, le flexing permet d'évaluer l'impact des variations dans les hypothèses sur les projections financières, tandis que la modélisation intégrée vise à analyser de manière cohérente et globale la relation entre les actifs et les passifs.

2.7 Présentation du bilan initial de l'assureur étudié

Actif		**Passif**	
Obligations	135 227 567 €	Capitaux Propres	900 000 €
Actions	27 045 513 €	Provisions mathématiques	173 088 479 €
Immobiliers	9 015 171 €	Provision pour participation aux bénéfices	6 000 000 €
Cash	8 700 227 €		
Total	179 988 479 €	Total	179 988 479 €

Table 11 – Bilan en valeur comptable de l'assureur au cout historique

Actif	
Obligations	119 110 964 €
Actions	32 454 616 €
Immobiliers	10 818 205 €
Cash	8 700 227 €
Total	171 084 013 €

Table 12 – Actif en valeur de marché de l'assureur

Ainsi, à l'initialisation, les obligations sont en moins-values, les actions et l'immobilier sont en plus-values. Le portefeuille obligataire affiche un taux de rendement actuariel moyen de 2 %, avec une duration moyenne d'environ huit ans. Il se compose de 36 % d'obligations d'entreprises et de 64 % d'obligations d'État.

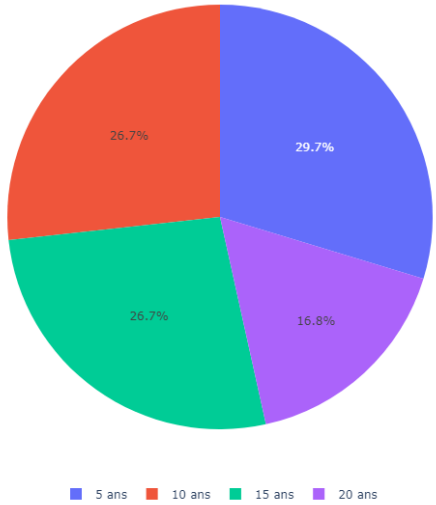

FIGURE 14 – Répartition des obligations par maturité

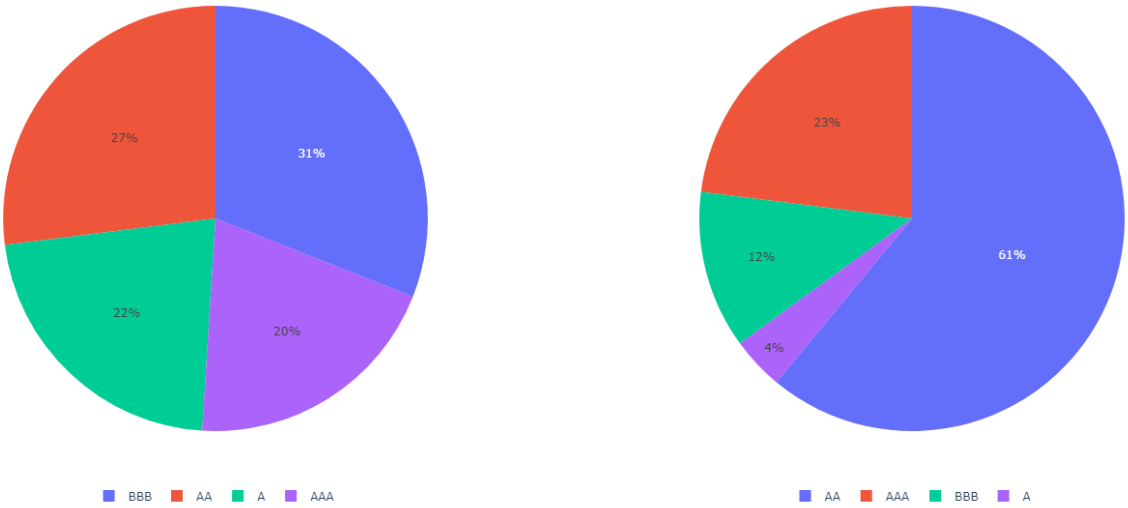

FIGURE 15 – Répartition par rating des obligations corporate (gauche) et souveraines (droite)

Rappel Dans le cadre du mémoire, l'objectif est d'optimiser le ratio $\frac{NAV}{SCR}$, la NAV (net asset value) inclut le risk margin.

3 Machine learning

3.1 Définition de l'environnement

L'objectif de ce mémoire est de proposer un procédé afin d'optimiser l'allocation cible d'actifs pour une compagnie d'assurance-vie. L'étude scindera les actifs en trois classes :

- Les obligations
- Les actions d'entreprise
- Les actifs immobiliers

Il est à noter que ce mémoire traite de l'allocation cible et non de l'allocation initiale de l'assuré. L'allocation initiale est celle définie au tableau 11.

3.1.1 Fonction objectif

Pour imposer une relation d'ordre entre les différents actifs, une fonction objectif doit être définie. Cette fonction objectif se décompose en trois parties :

- Un élément de rendement du portefeuille en scénario central : la net asset value (NAV)
- Un élément de risque du portefeuille : le SCR marché
- Un élément pour contrôler la dynamique du portefeuille en scénario central : Le best estimate liabilities (BEL)

$$\max_{x\in\mathcal{E}} \frac{\mathrm{NAV}_{\mathrm{Central}}(x)}{\mathrm{SCR}_{\mathrm{marché}}(x)} + \alpha_{\mathrm{dynamique}} e^{-\|f(\mathrm{BEL}_{\mathrm{central}}) - f(\mathrm{BEL}_{\mathrm{objectif}})\|_2^2} \qquad \text{(Fonction objectif)}$$

où :

- f est une fonction de description de l'évolution du BEL : par exemple, la courbure, la vitesse de décroissance...
- $\alpha_{\mathrm{dynamique}}$ est l'appétence à la contrainte dynamique
- $\mathcal{E}$ est l'espace possible de l'allocation d'actif.

Rappel Pour calculer le SCR marché, les chocs appliqués sur les taux, les spreads, l'immobilier et les actions seront pris en compte. Les risques de devise et de concentration ne seront pas pris en compte.

3.1.2 Description de l'espace des allocations possibles

L'espace des allocations cibles possibles est dénommé $\mathcal{E}$.

$$\mathcal{E} = \{(x, y, z) | x + y + z \leq 1, (x, y, z) \in [0, 1]^3\} \tag{$\mathcal{E}$}$$

Il s'agit donc d'un espace de dimension trois, car trois types d'actifs sont identifiés. Ensuite, pour quantifier cette espace, x, y, z sont supposés avoir une précision à l'échelle du dixième de pourcent. Ainsi, l'espace $\mathcal{E}$ est fini et se décompose en 176 845 points.

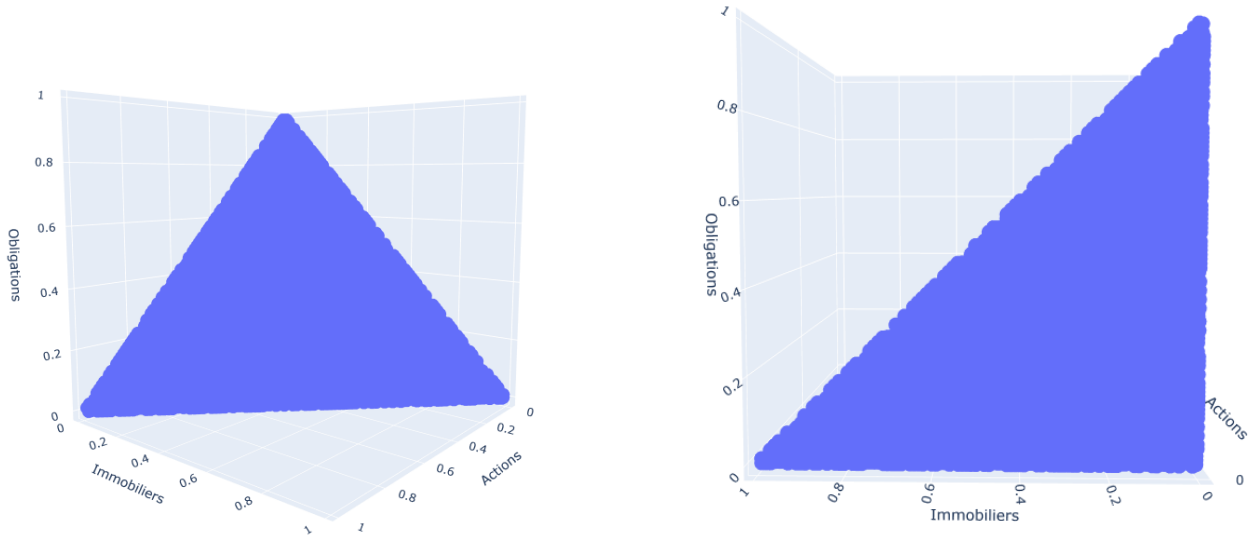

FIGURE 16 – Visualisation de l'espace des allocations cibles possibles

Intérêt du machine learning Pour estimer l'allocation d'actif optimale sans machine learning et en procédant séquentiellement, il faudrait compter 10 ans en calcul séquentiel [1] pour explorer l'ensemble de l'espace. En optimisant avec des calculs en parallèle pour la mesure du SCR marché, le délai serait encore supérieur d'un an et demi [2]. En utilisant des fonctions d'approximation à l'aide du machine learning et des méthodes d'exploration intelligentes, l'objectif est de réduire considérablement ce délai pour permettre aux compagnies d'assurance de définir leur allocation cible.

3.1.3 Stratégie

Contraintes Les modèles travaillent à environnement économique fixe. Ensuite, l'un des enjeux est le calcul des sorties du modèle ALM. En effet, une simulation avec 1000 scénarios nécessite cinq minutes. Ainsi, il est nécessaire d'avoir 7 fonctions d'approximation pour estimer le SCR marché et ajouter une contrainte dynamique :

- approximation pour la NAV centrale
- approximation en scénario central pour la caractérisation dynamique du BEL
- approximation pour la NAV actions
- approximation pour la NAV immobilier
- approximation pour la NAV spread
- approximation pour la NAV taux à la hausse
- approximation pour la NAV taux à la baisse

1. 5 min pour un module de risque composé de 500 scénarios économiques sur un ordinateur portable, 6 modules et 176 845 allocations à tester
2. En parallélisant les sous modules, le temps est divisé par 6

Architecture Pour constituer une base de données de taille suffisante, le(s) modèle(s) prendront au minimum en input les scénarios économiques et les allocations cibles. Trois architectures sont identifiables.

1. La première architecture consiste à associer à chaque sous-module un modèle de machine learning pour estimer le calcul de la NAV associée.
2. La deuxième architecture est de construire un modèle unique plus général qui a pour entrée supplémentaire une variable désignant le sous-module pour lequel la prédiction est faite. Cette approche permet aux autres modules de bénéficier des apprentissages effectués sur leurs voisins. Ce modèle unique aura une complexité plus grande.
3. Une architecture hybride : certaines fonctions à approximer utilisent la deuxième architecture, tandis que d'autres sont isolées en utilisant la première architecture.

Dans le cadre de ce mémoire, la troisième architecture sera étudiée, forçant ainsi l'implémentation et l'étude des deux précédentes afin d'obtenir la meilleure précision.

Méthode d'apprentissage et d'exploration Pour rappel, le coût temporel pour obtenir des outputs du modèle est très élevé.

Pour entraîner ces architectures, la première idée est d'échantillonner l'espace des allocations et des scénarios économiques de chacun des modules. Ensuite, à l'aide du machine learning à partir de quelques points, l'ensemble de l'espace est estimé.

La seconde idée est de travailler sur l'exploration et l'apprentissage simultanés du modèle. À chaque itération, un agent décide des scénarios, de l'allocation et des sous-modules à entraîner afin de maximiser la fonction objectif et la précision du système.

Ensuite, pour vérifier la solution proposée, des runs sur le modèle ALM sont lancés pour estimer les différents indicateurs réglementaires et vérifier que l'allocation identifiée est meilleure que celle actuelle.

Dans le cadre de ce mémoire, compte tenu de la limite des outils à disposition, la première méthode sera utilisée pour optimiser l'allocation cible d'actifs. Pour ce mémoire, un échantillonnage par maximisation de la diversité est utilisé. Il permet la construction d'une base de données pour chaque sous-module du SCR marché. Une fois la base générée, il n'y a plus besoin d'utiliser le modèle ALM codé sur Prophet. Ensuite, cette méthode facilite l'optimisation des hyperparamètres.

3.2 Construction de la base de données

3.2.1 Caractérisation des scénarios économiques

Chaque scénario économique est caractérisé par 3 264 variables. Les scénarios économiques pour le calcul du SCR marché se scindent en trois catégories :

— Le scénario central
— Le scénario à taux bas
— Le scénario à taux haut

L'ensemble de ces variables ne sont pas nécessaires pour caractériser un scénario. Ainsi, pour concentrer l'information, un autoencodeur est développé (voir 3.2.1).

Base de donnée Pour développer une fonction de réduction de dimension, l'étude s'effectuera sur 5 000 scénarios univers central, 5 000 scénarios univers taux haut et 5 000 scénarios univers taux bas.

Analyse des scenarios via Analyse en composante principale L'analyse en composante principale ne permet pas de distinguer efficacement les différences entre les scénarios. De plus, pour atteindre une explication de l'information de 95%, il faudrait plus de 20 variables (figure 17), dont certaines avec un intérêt très faible par rapport aux premières. Ainsi, pour obtenir des variables avec une capacité d'explication, cette étude a fait le choix d'utiliser un autoencodeur (voir 3.2.1).

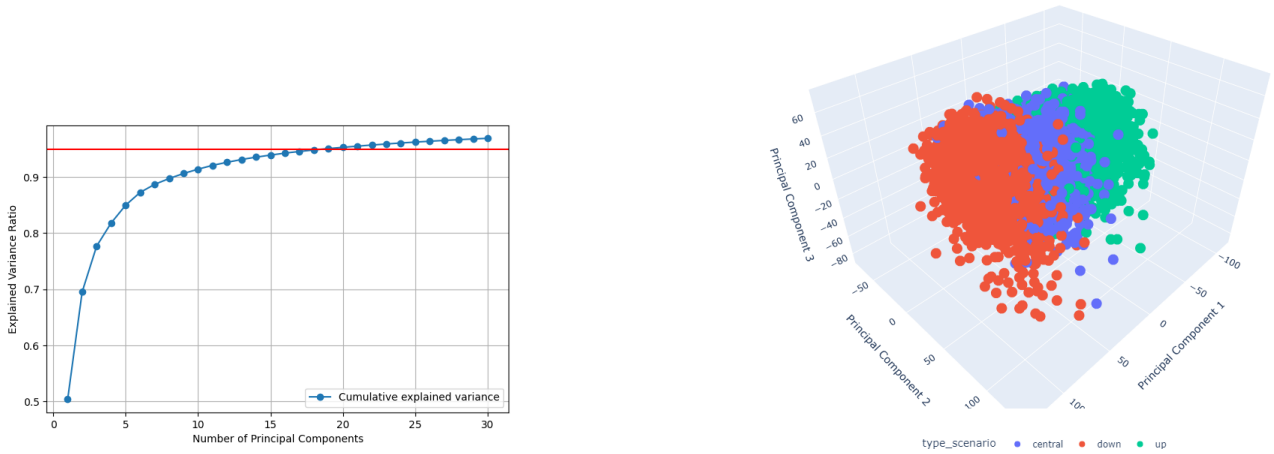

FIGURE 17 – Variance cumulée expliquée (gauche) et visualisation du dataset avec PCA (droite)

Conception de l'autoencodeur Un autoencodeur est une architecture de réseau de neurones qui se compose en deux parties (voir figure 18) :

— un encodeur
— un décodeur

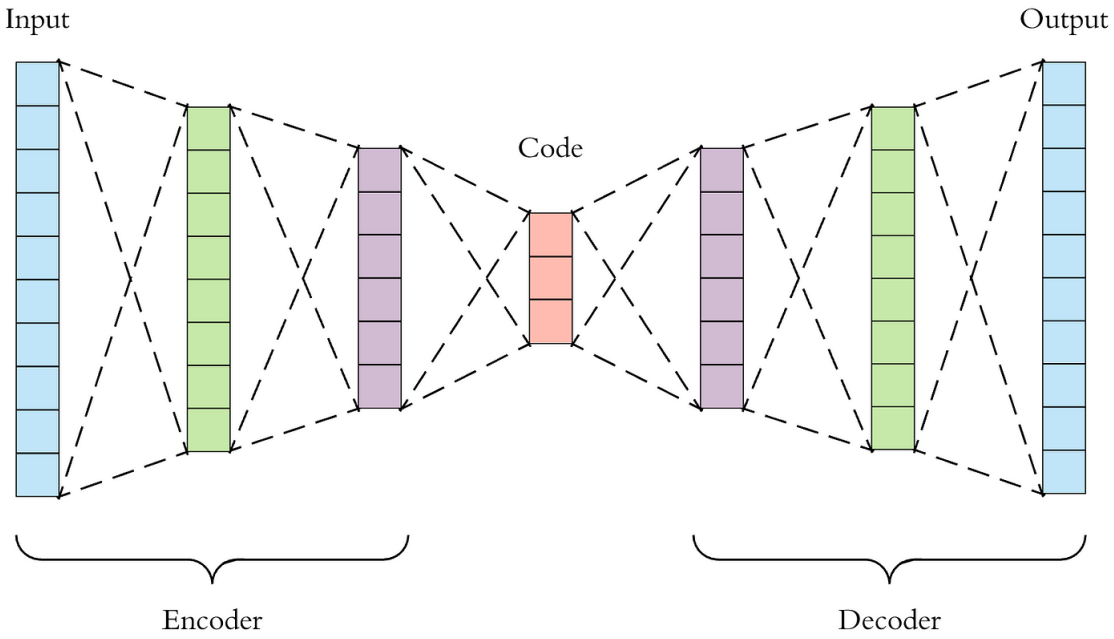

FIGURE 18 – Schéma de l'architecture d'un autoencodeur

Dans le cadre du mémoire, l'encodeur et le décodeur auront une structure symétrique. La taille de l'espace d'encodage sera de 128 nœuds. La taille ainsi que les fonctions d'activation sont déterminées en utilisant une technique d'optimisation des hyperparamètres via la bibliothèque Optuna.

Best MSE	0.00887
Nombre de couches décodeur	2
Nombre de couches encodeur	2
Fonction activation	selu
encodeur taille couche 1	224
encodeur taille couche 2	192
décodeur taille couche 1	192
décodeur taille couche 2	224

TABLE 13 – Meilleurs paramètres

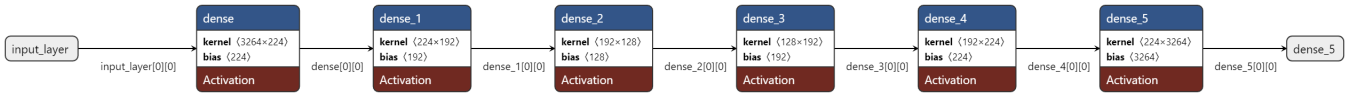

FIGURE 19 – Décomposition des couches de l'autoencodeur

Cette variation brusque de taille entre l'entrée et la première couche de l'encodeur s'explique par le fait que les 3 264 variables sont liées entre elles. L'information contenue est très redondante. Ainsi, le modèle possède 1 601 664 paramètres. Pour l'entraînement de l'autoencodeur, les données d'entrées sont pré-traitées avec un Standard Scaler.

$$z = \frac{x - \mu}{\sigma}$$

où μ est la moyenne de l'échantillon, σ l'écart-type de l'échantillon. L'ensemble de la base de données est découpé pour avoir 10% en test et 90% en train. La batch size est fixée à 32 et deux callbacks ont été utilisés :

— un Earlystopping pour un nombre maximum d'epochs de 140 :

Listing 1 – Code tensorflow pour le callback Earlystopping

```
tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=10,
    verbose=1,
    restore_best_weights=True
)
```

— Un Reduce learning rate on plateau avec un learning rate initial de 0.001 :

Listing 2 – Code tensorflow pour le callback Reduce learning rate on plateau

```
tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.2,
    patience=5,
    min_lr=1e-8,
    verbose=1
)
```

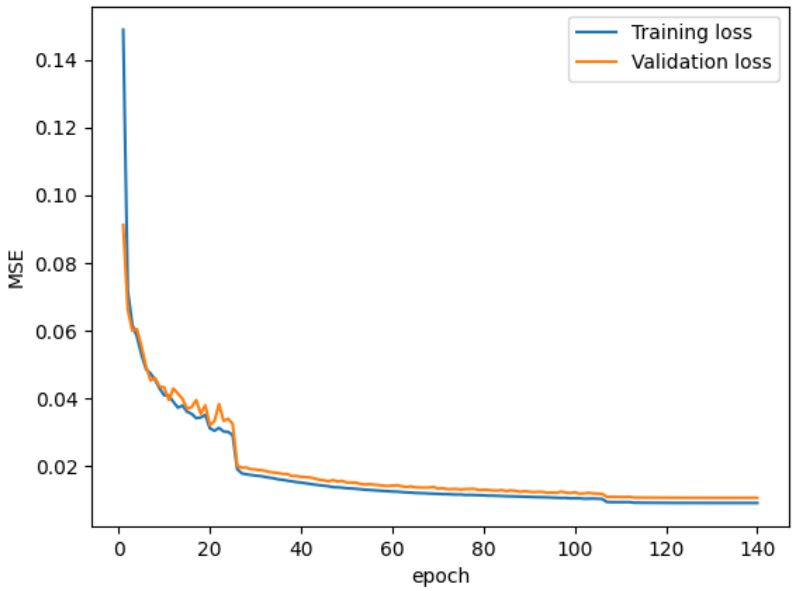

FIGURE 20 – apprentissage autoencodeur : loss pour les datasets d'entrainement et de validation par epoch

La courbe d'évolution de la loss (20) témoigne de l'utilité du callback Reduce learning rate on plateau. En effet, des décrochages sont présents pendant la phase d'apprentissage. De plus, la diminution a permis de stabiliser la validation loss.

Visualisation de l'encodade Pour visualiser l'encodage réalisé, les algorithmes t-SNE et UMAP

- **t-SNE (t-distributed Stochastic Neighbor Embedding)** : Cette méthode convertit les distances euclidiennes entre points de données en probabilités de similarité conditionnelle, et cherche à reproduire ces probabilités dans un espace de dimension réduite (généralement 2D ou 3D) en minimisant la divergence de Kullback-Leibler entre les distributions de similarité. t-SNE est efficace pour conserver les structures locales et est particulièrement utile pour identifier des clusters.
- **UMAP (Uniform Manifold Approximation and Projection)** : Cette méthode vise à conserver les structures locales et globales des données. Basé sur des théories de géométrie algébrique et topologique, UMAP construit un graphe de voisinage des données et optimise une fonction de coût pour maintenir cette structure lors de la projection dans un espace de plus faible dimension.

FIGURE 21 – Visualisation de l'encodage : UMAP à gauche, t-SNE à droite
(rouge : taux bas, bleu : taux central, rose : taux haut)

Les figures 21 montrent la capacité de l'autoencodeur à distinguer les scénarios en univers central, taux haut et taux bas. En conclusion, l'autoencodeur est capable de distinguer clairement les scénarios en taux haut. En revanche, un point de vigilance est à noter pour les scénarios en taux bas et central : UMAP ne fait pas clairement la distinction entre ces deux univers. Plusieurs sources d'amélioration sont identifiables : la calibration sur XSG pour la génération des scénarios taux bas ou central aurait pu être plus adaptée ou l'apprentissage de l'autoencodeur est insuffisant. Dans le cadre des objectifs du mémoire, l'autoencodeur sera considéré comme satisfaisant avec une erreur quadratique moyenne de 0.00887. En effet, l'autoencodeur a été développé afin d'améliorer l'échantillonnage. Selon la figure 21, l'autoencodeur est en capacité de proposer un échantillonnage cohérent de l'espace des scénarios économiques. En revanche, une réserve doit être émise quant à sa capacité à distinguer avec un haut niveau de confiance les scénarios suivant leur type (central, taux haut, taux bas). Dans le cadre de ce mémoire, seul le premier point est essentiel, puisqu'un échantillonnage différent est réalisé pour chaque type de scénario.

Vérification des valeurs de Shap Les valeurs de Shap permettent de compléter l'analyse de l'autoencodeur (figure 22) et de justifier son intérêt par rapport à une analyse en composante principale. Sur l'analyse en composante principale, l'intérêt des variables diminue rapidement. Alors qu'ici, la valeur de Shap de chaque feature ne diminue pas de manière brutale. Ainsi, le modèle développé est capable d'apprendre des relations plus complexes que celles proposées par PCA.

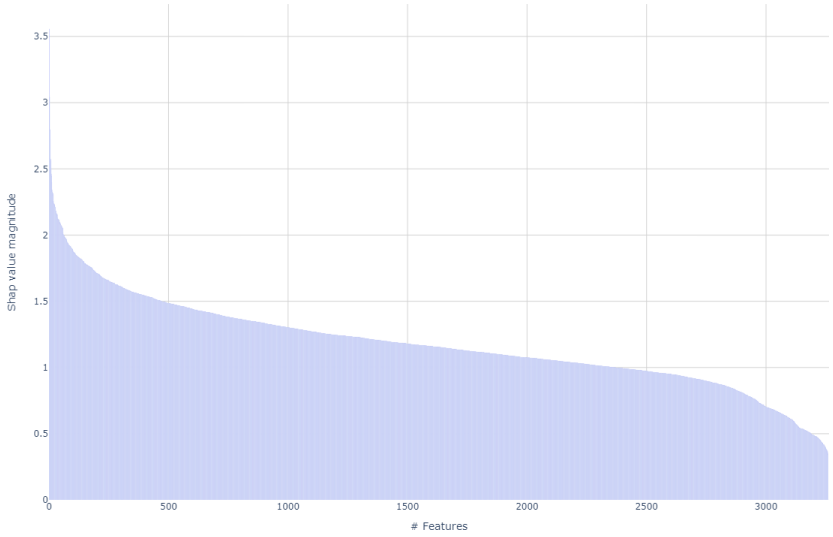

FIGURE 22 – Magnitude moyenne des valeurs de Shap

3.2.2 Développement de l'interface de communication entre les différents outils

L'enjeu de ce mémoire concerne l'utilisation du modèle ALM sur Prophet. La problématique étudiée nécessite de réaliser un très grand nombre de simulations (de l'ordre du million). De plus, chaque simulation possède des caractéristiques particulières. Alors, une partie du mémoire est consacrée à l'automatisation et au contrôle des processus Prophet depuis python. Au-delà du cadre du mémoire, ce travail présente une réelle valeur pour l'ensemble de l'équipe actuariat, car le développement de cette connexion directe entre python et Prophet permettra de simplifier et de réduire les risques des processus actuels. Aujourd'hui, le contrôle de Prophet depuis python se fait en utilisant des fichiers Excel pour lancer des commandes VBA.

Procédure Pour connecter, l'environnement python directement à Prophet, les étapes suivantes sont réalisées :

1. Migration de l'environnement python du 64 bit vers le 32 bit pour s'aligner sur le fonctionnement de Prophet
2. Récupération du fichier contenant l'ensemble des commandes Prophet utilisées par VBA
3. Conversion du fichier en bibliothèques python en utilisant com types
4. Création de classes afin de gérer le fonctionnement Prophet depuis python

3.2.3 Sélection des inputs et des outputs

Inputs Le modèle développé a besoin des informations suivantes :

- une variable à 6 états pour caractériser le sous-module du SCR marché
- un ensemble de variables descriptives des scénarios économiques
- 3 variables pour caractériser l'allocation des actifs parmi les points situés sur la figure 16

Ouputs Le système proposé se compose donc de deux modèles, l'un plus complexe pour prédire les NAVs des différents sous-modules. Ensuite, un second chargé de prédire les variables caractéristiques du BEL. Pour rappel, le premier modèle est plus complexe, car ce modèle navigue entre les différents sous-modules du SCR marché ; tandis que le second reste en scénario central.

3.3 Échantillonnage de l'espace

Pour échantillonner les différents espaces (les scénarios économiques et les allocations), la méthode d'échantillonnage par maximum de diversité (Maximim Sampling) est utilisée. Cette technique vise à sélectionner des points de manière à maximiser la distance minimale entre les points échantillonnés. Cela garantit que les points sont bien dispersés dans l'espace, représentant ainsi mieux les différentes régions de l'espace de haute dimension. Cette méthode a été choisie, car elle assure une bonne couverture de l'espace et évite la sélection de points trop proches les uns des autres. De plus, elle est plus efficace pour des espaces de haute dimension où des méthodes comme l'échantillonnage aléatoire simple peuvent ne pas suffire.

Listing 3 – Code pour l'échantillonnage par maximum de diversité

```python
import numpy as np
from scipy.spatial.distance import cdist

def maximin_sampling(points, n):
    selected_points = [points[np.random.choice(points.shape[0])]]
    for _ in range(1, n):
        dists = cdist(points, selected_points)
        min_dists = dists.min(axis=1)
        next_point = points[np.argmax(min_dists)]
        selected_points.append(next_point)
    return np.array(selected_points)
```

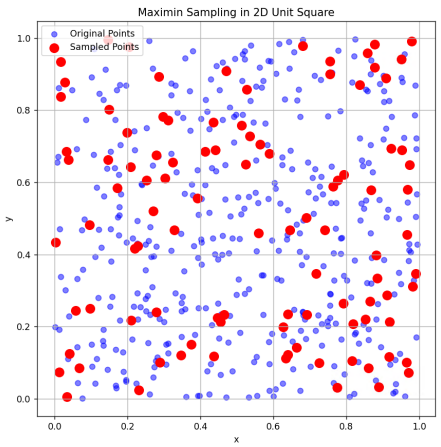

FIGURE 23 – Exemple échantillonnage avec Maximim (500 points et 100 points sélectionnés)

3.3.1 Construction de la fonction de caractérisation du comportement dynamique du BEL

Pour contrôler la dynamique d'évolution des engagements au passif, une caractérisation de l'évolution du BEL est nécessaire. Comme illustré par la figure 24, d'un scénario à l'autre, la courbe peut grandement évoluer.

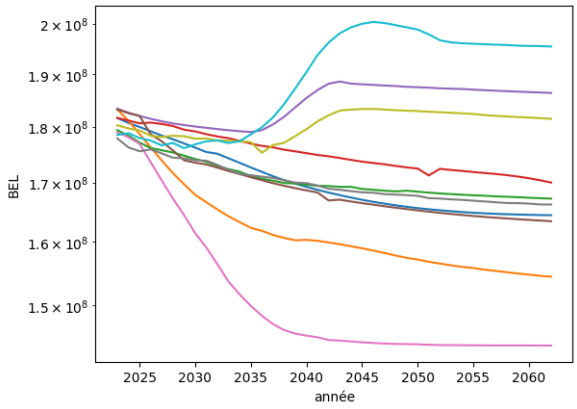

FIGURE 24 – Exemple de Profil du BEL en fonction du temps

Pour définir cette fonction, une analyse en composante principale est réalisée. Pour construire une base de donnée afin d'extraire un nombre limité de features, pour chaque scénario du scénario central, la courbe caractérisant le BEL en fonction du temps sera extraite.

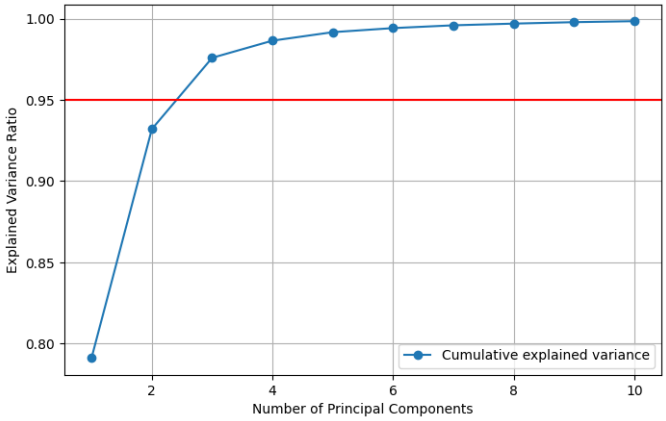

FIGURE 25 – Variance expliquée par la PCA des profils de BEL en fonction du temps

La méthode de la PCA est suffisante pour caractériser les BELs. En effet, avec 3 composantes, la variance expliquée cumulée dépasse 95 % (figure 25). De plus, sur la visualisation, plusieurs hyperplans et clusters se distinguent, ce qui prouve l'efficacité de la PCA (figure 26).

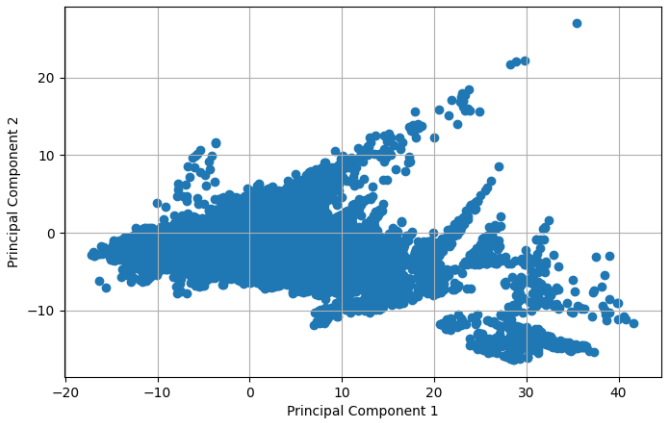

FIGURE 26 – Visualisation d'un échantillon des profils de BEL avec la PCA

3.4 Typologie des modèles de machine learning

Dans le cadre de ce mémoire, plusieurs types de modèles de machine learning seront étudiés et combinés *in fine* afin de bénéficier des avantages de chacun et de décider lesquels seront les plus intéressants. Ainsi les modèles suivants seront présentés :

1. Xgboost pour les arbres aléatoires
2. Dense neural network (DNN) pour le deep learning
3. Kernel Ridge regression pour les régressions

3.4.1 XGBOOST

Introduction à Xgboost XGBoost (eXtreme Gradient Boosting) est une implémentation avancée des arbres de décision en ensemble. C'est un algorithme d'apprentissage supervisé très populaire en raison de ses performances et de sa flexibilité. Il est particulièrement efficace pour les problèmes de régression et de classification.

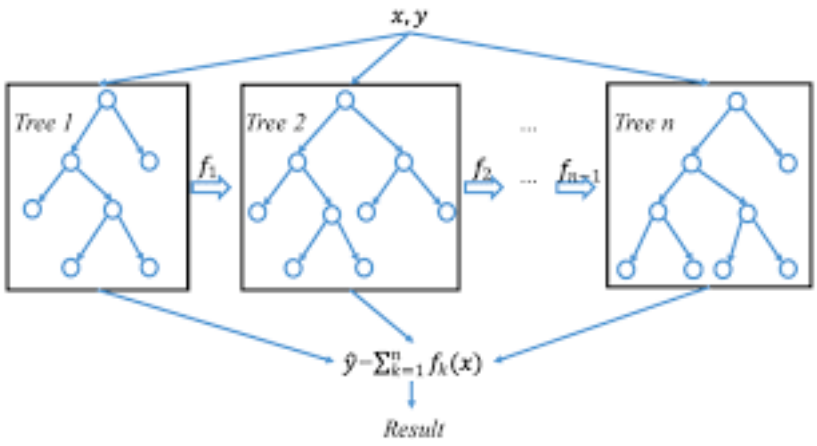

FIGURE 27 – Architecture du modèle XGBoost

Fonctionnement XGBoost est basé sur le principe de boosting, qui combine plusieurs modèles faibles (typiquement des arbres de décision de faible profondeur) pour former un modèle puissant. Le boosting itératif ajuste successivement les erreurs des modèles précédents. Ainsi, un modèle XGBoost suit les étapes suivantes :

1. **Initialisation :** XGBoost commence par la construction d'un modèle de base simple. Traditionnellement, ce modèle de base est un arbre de décision initiale peu profond, et les prédictions initiales peuvent être une valeur constante, souvent choisie comme la moyenne des valeurs cibles dans le cas de la régression. Cette étape est cruciale, car elle fournit un point de départ pour l'algorithme de boosting, à partir duquel les erreurs résiduelles seront calculées et corrigées. L'objectif est d'établir une base initiale solide qui pourra être améliorée à chaque itération suivante.
2. **Construction d'arbres successifs :** Une fois le modèle de base établi, XGBoost procède à la construction d'arbres de décision successifs. Chaque nouvel arbre est formé pour corriger les erreurs commises par les arbres précédents. À chaque itération, les résidus (ou erreurs) des prédictions précédentes sont utilisés comme nouvelles cibles pour l'arbre en cours de construction. Cela signifie que chaque nouvel arbre se concentre spécifiquement sur les observations mal prédites précédemment, améliorant progressivement la précision globale du modèle. Cette approche itérative permet de réduire les erreurs de manière efficace en se concentrant sur les aspects les plus difficiles du problème.
3. **Optimisation :** L'optimisation dans XGBoost repose sur la minimisation d'une fonction objective qui intègre à la fois une fonction de perte et un terme de régularisation. La fonction de perte mesure l'écart entre les prédictions du modèle et les valeurs réelles, par exemple l'erreur quadratique moyenne pour les problèmes de régression. Le terme de régularisation, quant à lui, contrôle la complexité du modèle pour éviter le sur-apprentissage (overfitting), en pénalisant les modèles trop complexes. Les hyperparamètres tels que le taux d'apprentissage, la profondeur maximale des arbres et le nombre minimal de données dans une feuille jouent un rôle crucial dans cette optimisation. Ils doivent être ajustés avec soin pour équilibrer la précision et la généralisation du modèle.
4. **Importance des features :** Une des fonctionnalités puissantes de XGBoost est sa capacité à évaluer l'importance des caractéristiques. Lors de la construction des arbres, XGBoost mesure l'impact de chaque caractéristique sur la réduction de la fonction de perte. Cette évaluation se fait en analysant la contribution de chaque variable explicative aux décisions prises par les arbres successifs. Les caractéristiques qui contribuent le plus à la réduction de la perte, sont considérées comme les plus importantes. Cette information est précieuse pour les actuaires, car elle permet de comprendre quelles variables influencent le plus les résultats et ainsi d'ajuster les modèles en conséquence.

Recherche des hyperparamètres : À l'aide du module d'optimisation Optuna, les hyperparamètres suivants seront étudiés :

- **n_estimators** : Correspond au nombre d'arbres à entraîner. Une valeur trop basse peut entraîner un sous-ajustement (underfitting), tandis qu'une valeur trop élevée peut entraîner un sur-ajustement (overfitting).
- **learning_rate** : Aussi appelé taux d'apprentissage, cet hyperparamètre contrôle la contribution de chaque arbre. Un taux plus bas rend l'entraînement plus lent, mais peut améliorer les performances en permettant des ajustements plus fins.
- **max_depth** : Détermine la profondeur maximale des arbres. Des arbres plus profonds peuvent modéliser des relations plus complexes, mais risquent aussi de sur-ajuster les données.
- **min_child_weight** : Cet hyperparamètre contrôle la taille minimale de l'ensemble d'observations (poids de l'enfant) nécessaire dans un nœud. Des valeurs plus élevées empêchent la modélisation de relations très spécifiques (overfitting).
- **subsample** : Indique la fraction des données utilisées pour entraîner chaque arbre. Des valeurs plus basses peuvent aider à prévenir le sur-ajustement.

— **colsample_bytree** : Cet hyperparamètre spécifie la fraction de caractéristiques à utiliser pour chaque arbre. Il aide également à prévenir le sur-ajustement.
— **lambda** : Aussi appelé régularisation L2, cet hyperparamètre ajoute une pénalité pour les coefficients larges, ce qui aide à prévenir le sur-ajustement.
— **alpha** : Aussi appelé régularisation L1, cet hyperparamètre ajoute une pénalité pour le nombre de coefficients non nuls, favorisant ainsi des modèles plus simples et moins susceptibles de sur-ajuster.

Listing 4 – Code pour l'entrainement d'un modèle XGBoost

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
scaler = StandardScaler()
y_train = scaler.fit_transform(y_train.reshape(-1, 1)).ravel()
y_test = scaler.transform(y_test.reshape(-1, 1)).ravel()

param = {
    'objective': 'reg:squarederror',
    'n_estimators': trial.suggest_int('n_estimators', 50, 1000),
    'learning_rate': trial.suggest_float('learning_rate', 1e-6, 0.3),
    'max_depth': trial.suggest_int('max_depth', 2, 10),
    'min_child_weight': trial.suggest_int('min_child_weight', 1, 10),
    'subsample': trial.suggest_float('subsample', 0.5, 1.0),
    'colsample_bytree': trial.suggest_float('colsample_bytree', 0.5, 1.0),
    'lambda': trial.suggest_float('lambda', 1e-8, 1.0, log=True),
    'alpha': trial.suggest_float('alpha', 1e-8, 1.0, log=True),
}
model = xgb.XGBRegressor(**param)
model.fit(X_train, y_train, eval_set=[(X_test, y_test)])
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
```

Premiers résultats et mise à jour de la stratégie Pour constituer le modèle de référence, un modèle XGBoost avec l'ensemble des variables des scénarios économiques et l'allocation a été testé. L'erreur relative issue du modèle est supérieure à 300 % sur le test et le train set. Pour mieux comprendre les raisons de ce niveau d'erreur, plusieurs points sont à étudier :
— L'apprentissage du modèle
— La corrélation entre les features et l'output

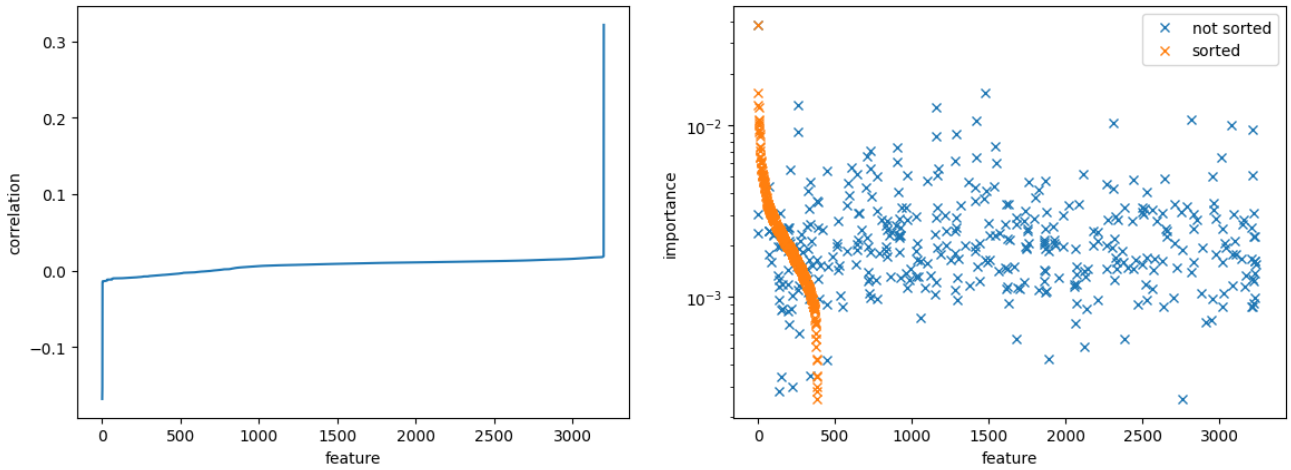

FIGURE 28 – Corrélation entre les features et la sortie (gauche) et l'importance de chaque input pour le modèle (droite)

La figure 28 montre que le système est noyé dans un environnement avec des entrées sans lien réel avec la sortie. Après analyse, le modèle utilise seulement quelques features pour faire ces prédictions. En effet, le modèle utilise essentiellement les trois variables caractérisant l'allocation pour faire ses prédictions. Les figures 29 montrent que la NAV évolue dans un premier temps en suivant la proportion d'obligation. De ces observations se dégagent deux conclusions :

— L'allocation est une variable clef pour comprendre l'évolution de la NAV
— L'allocation a une telle importance qu'elle noie les autres features, empêchant le modèle d'apprendre correctement.

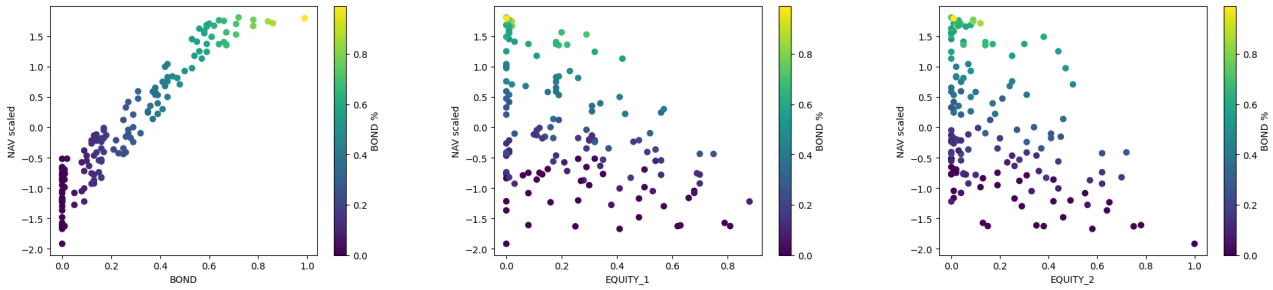

FIGURE 29 – Évolution de la NAV en fonction de l'allocation

Solution 1 Pour vérifier la compréhension du problème et le résoudre, une première idée est de modifier l'input du modèle. Pour rappel, précédemment, un data point était la combinaison de l'allocation et du scénario économique. Désormais, au vu de l'importance de l'allocation, une moyenne sur l'ensemble des scénarios économiques est faite par allocation. Suite à l'entraînement, l'erreur obtenue est de l'ordre de 25 % avec un dataset de 150 lignes.

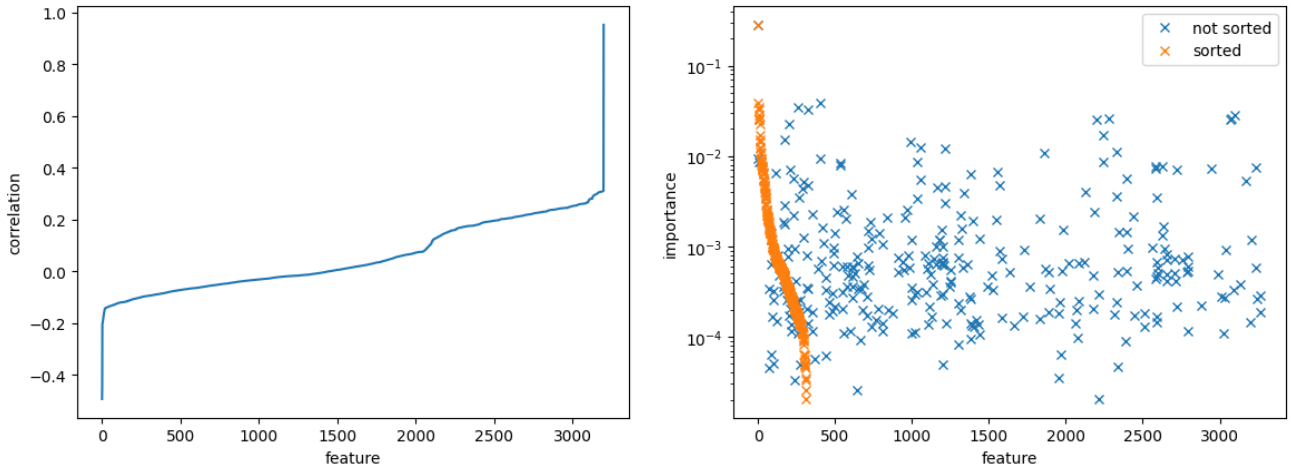

FIGURE 30 – Corrélation entre les features et la sortie (gauche) et l'importance de chaque input pour le modèle (droite)

L'erreur s'est nettement réduite. La corrélation entre les inputs et les outputs est également améliorée, mais l'allocation reste prépondérante. Il en va de même pour l'importance des features.

Solution 2 Une deuxième idée est d'utiliser l'encodeur pour condenser l'information à l'aide de l'autoencodeur.

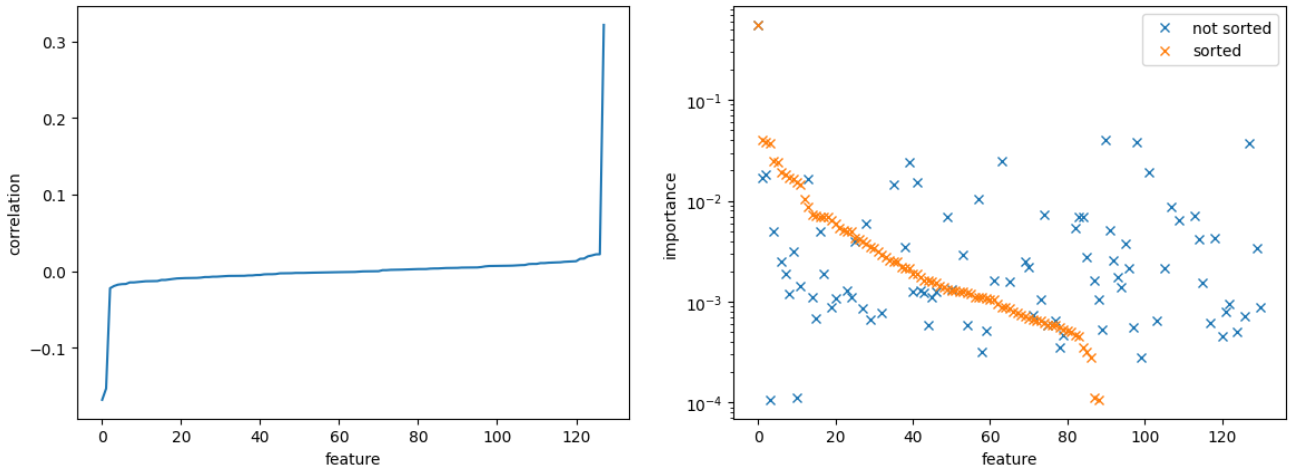

FIGURE 31 – Corrélation entre les features et la sortie (gauche) et l'importance de chaque input pour le modèle (droite)

L'erreur relative obtenue est de l'ordre de 300 %. L'importance accordée à chaque feature est améliorée, mais comme pour le premier modèle, les variables d'allocation écrasent celle du scénario, empêchant le modèle d'apprendre correctement (figure 31).

Solution 3 Une troisième idée est de combiner les deux solutions.

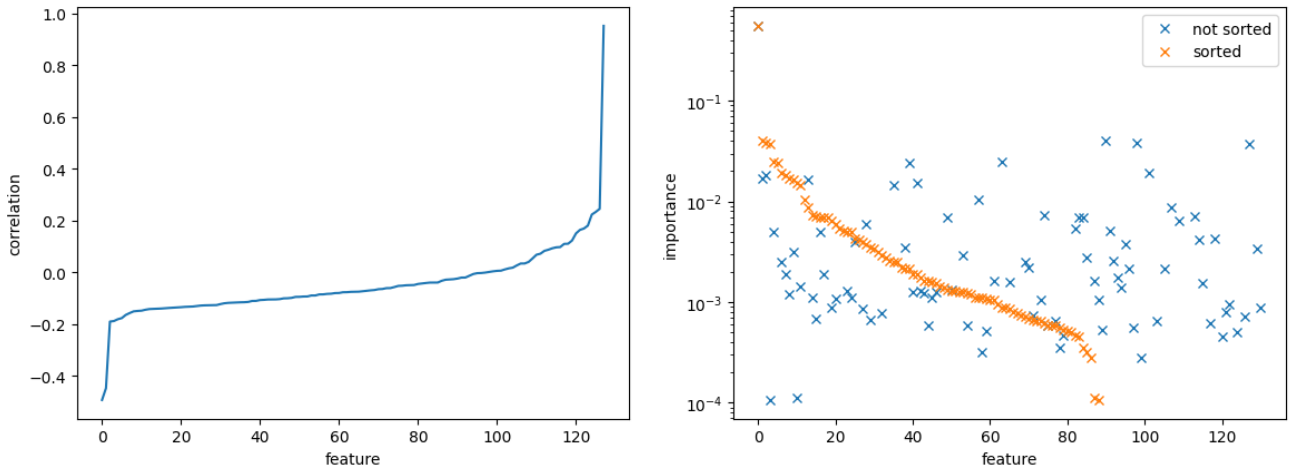

FIGURE 32 – Corrélation entre les features et la sortie (gauche) et L'importance de chaque input pour le modèle (droite)

Une légère amélioration sur l'erreur relative (23 %), mais à contrebalancer avec la faible taille du dataset. La corrélation entre les inputs et les outputs est améliorée, et l'importance des features est inchangée.

Bilan Pour entraîner correctement un modèle, deux options sont possibles.

1. La première est la construction d'une suite de variables permettant au système de prendre en compte le scénario économique dans sa prédiction. Il faut condenser davantage l'information contenue dans les scénarios économiques pour entrer en concurrence avec les variables de l'allocation. L'autoencodeur doit être amélioré pour permettre une meilleure compression de l'information.
2. La seconde, qui sera celle retenue car plus simple et plus certaine ; les variables économiques sont retirées de l'apprentissage et une moyenne est faite pour prédire la NAV du sous-module. Cette méthode impose de revoir les interactions avec Prophet pour s'affranchir du temps de compilation.

3.4.2 Neural Network

Le deep learning est une branche du machine learning. Il s'inspire du fonctionnement du cerveau humain. Ce type de modèle repose sur les principes suivants :

— Définition d'une structure : Les réseaux de neurones sont composés de couches de neurones. Chaque neurone possède un poids et une fonction d'activation. Il existe différents types de couches qui permettent de réaliser différentes interactions entre les différents neurones.

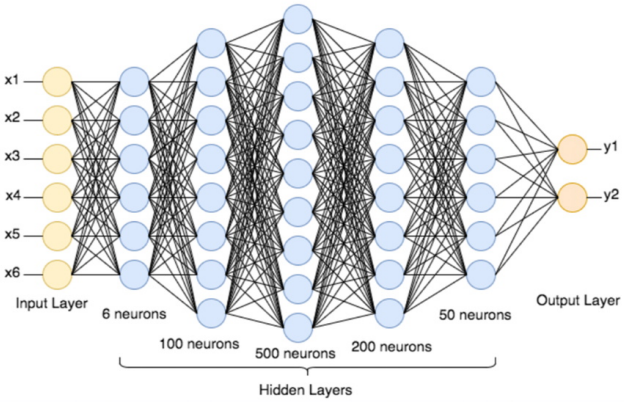

FIGURE 33 – Exemple : Structure d'un réseau dense

— Forward propagation : Cette étape permet à la donnée d'entrée de traverser l'ensemble des couches du réseau pour aboutir à une sortie (une prédiction).
— Fonction d'activation : Cette fonction propre à chaque couche permet d'ajouter de la non-linéarité. Cette non-linéarité fait la force des réseaux de neurones, car elle leur permet de capter des relations complexes.
— Fonction de coût : Cette fonction est celle que notre modèle va chercher à optimiser. Cette optimisation se fait à l'étape dite de Backpropagation. Le choix de la fonction coût est défini par le type de problème. Cette fonction doit idéalement être convexe pour avoir existence d'un minimum unique. Mean square error et cross entropy sont des exemples de fonction coût. Dans le cadre du challenge, la categorical cross entropy sera utilisée.
— Backpropagation : Cette étape met à jour les poids des neurones afin de minimiser la loss. Généralement, cette mise à jour des poids se fait à l'aide d'un algorithme de descente de gradient. Ici, les modèles seront entraînés à l'aide de la méthode Adam (adaptive moment estimation).

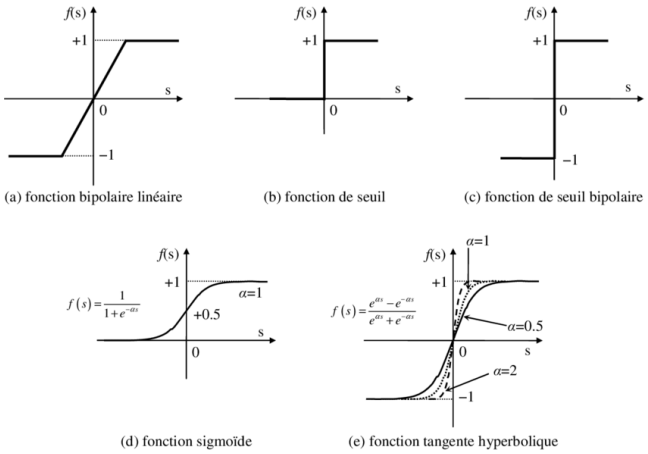

FIGURE 34 – Exemple de fonction d'activation

Dense Network Un réseau est dit dense lorsque la connexion entre les neurones de deux couches successives est complète. la sortie du i-ème neurone située à la couche j peut s'exprimer :

$$z_j = f(\sum_l w_{ij} a_j + b_j)$$

Dans le cadre de ce mémoire, des réseaux denses seront construits, car ils sont très performants en matière de régression non linéaire.

3.4.3 Kernel ridge régression

Pour le besoin du sujet d'étude et obtenir un modèle avec de meilleures capacités d'extrapolation, les régressions par noyau avec pénalisation Ridge sont présentées.

Régression Ridge La régression Ridge est une technique de régularisation qui est utilisée pour analyser des données présentant des multicolinéarités. Elle ajoute une pénalisation à la somme des carrés des coefficients de la régression linéaire, ce qui permet de réduire le surajustement du modèle.

La régression linéaire classique cherche à minimiser l'erreur quadratique :

$$\min_{\beta} \left(\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 \right)$$

La régression Ridge ajoute un terme de pénalisation :

$$\min_{\beta} \left(\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right)$$

où :
— y_i est la valeur réelle de la variable dépendante pour l'observation i,
— x_{ij} est la valeur de la variable explicative j pour l'observation i,
— β_0 est l'interception,
— β_j est le coefficient de la variable explicative j,
— λ est le paramètre de régularisation.
La solution de la régression Ridge est donnée par :

$$\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y$$

où :
— X est la matrice des variables explicatives,
— y est le vecteur des valeurs observées de la variable dépendante,
— I est la matrice identité.

Fonction noyau Les fonctions à noyaux sont des outils puissants en apprentissage automatique, permettant de transformer des données en un espace de caractéristiques de haute dimension pour capturer des relations non linéaires.

Des exemples classiques de fonctions à noyaux :
— **Noyau Linéaire** : Utilisé lorsque les données sont linéairement séparables.

$$K(x, x') = x^\top x' \tag{1}$$

— **Noyau Polynomial** : Capte des relations non linéaires de degré d.

$$K(x, x') = (1 + x^\top x')^d \tag{2}$$

— **Noyau Gaussien (RBF)** : Très flexible et capable de capturer des relations complexes.

$$K(x, x') = \exp(-\gamma \|x - x'\|^2) \tag{3}$$

— **Noyau Sigmoïde** : Utilisé dans certains types de réseaux de neurones.

$$K(x, x') = \tanh(\kappa x^\top x' + c) \tag{4}$$

Formulation mathématique de la régression Ridge avec noyau Soit une régression Ridge avec la fonction noyau K, l'estimateur $\hat{f}$ de la fonction f se note :

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i)$$

où les coefficients α_i sont trouvés en résolvant le système linéaire :

$$(K + \lambda I)\alpha = y$$

3.5 Analyse des données Prophet

3.5.1 Notations et vocabulaire

Dans les graphiques, pour coller aux notations de Prophet et alléger les légendes, les dénominations suivantes sont utilisées :

Nomination	Signification
BOND_1	Obligation
EQUITY_1	Action
EQUITY_2	Immobilier
sub_module_0	scénario central
sub_module_1	scénario taux choqué à la hausse
sub_module_2	scénario taux choqué à la baisse
sub_module_3	scénario actions choquées à la baisse
sub_module_4	scénario immobilier choqué à la baisse
sub_module_5	scénario spread choqué à la hausse

TABLE 14 – Légendes pour la nomination des variables

3.5.2 Corrélation entre features et cible

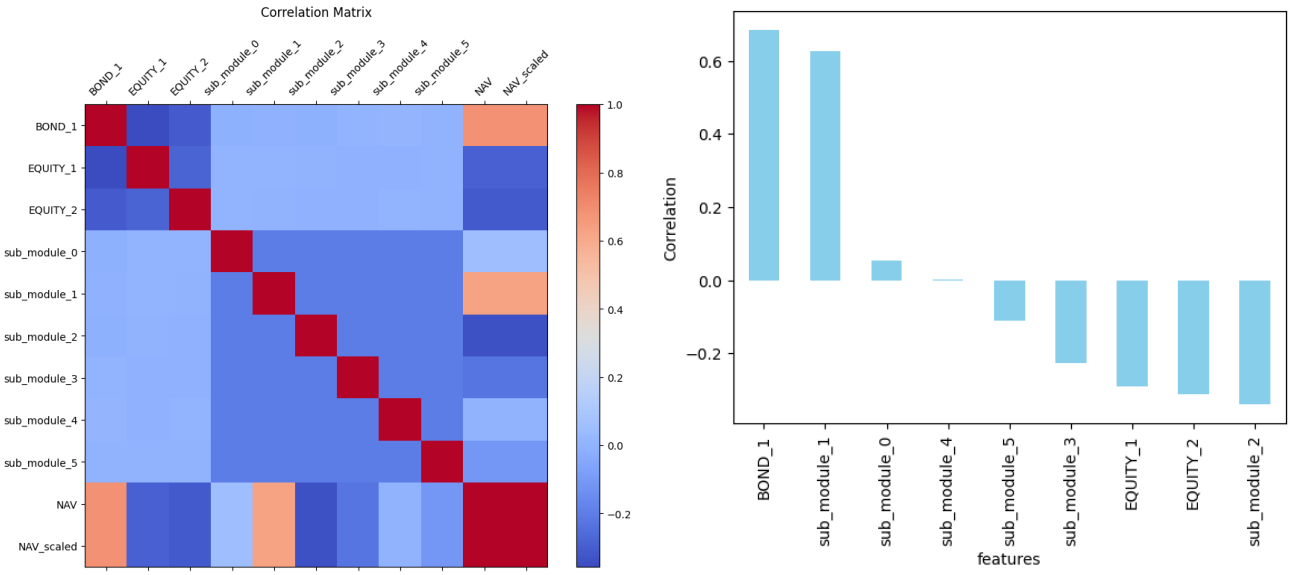

FIGURE 35 – Corrélation avec les données extraites de Prophet

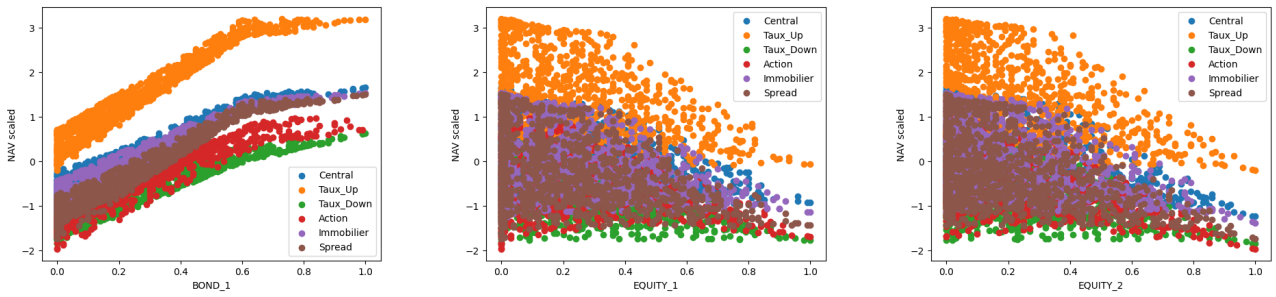

FIGURE 36 – Évolution de la NAV en fonction du portefeuille cible

La corrélation entre la NAV et les différents inputs mérite d'être expliquée d'un point de vue qualitatif.

- Au regard de la figure 36, la forte corrélation entre l'obligation et la NAV est triviale.
- Ensuite, il peut paraître étonnant de ne pas avoir la même corrélation pour les différents classifieurs des sous modules de risque. Cependant, en analysant la figure 36, cette différence de corrélation est expliquée : le sous-module "Taux haut" est très éloigné des autres sous-modules, ainsi la corrélation du classifieur "sub_module_1" (variable représentative des scénarios avec la courbe des taux choquée à la hausse) est importante.
- En raisonnant de façon similaire, la corrélation des autres classifieurs est expliquée.
- Ensuite, l'action et l'immobilier possèdent la même mesure de corrélation par rapport à la NAV, car elles évoluent de manière très similaire.

Dans le cadre d'une étude actuarielle, il est à noter que la NAV augmente lorsque la part en obligation du portefeuille augmente. Cette tendance s'explique par les éléments suivants :

- Simulation en risque neutre : le rendement des obligations est le même que celui de l'immobilier et des actions.
- Les garanties des contrats en euros et la participation aux bénéfices : le profit d'un assuré est asymétrique (forme d'un long call) par rapport au profit de l'assureur, seul l'assureur supporte la perte.

Ainsi, l'augmentation de la NAV avec la proportion d'obligations dans le portefeuille s'explique par la combinaison des deux effets ci-dessus. En raisonnant dans le sens inverse, en augmentant la proportion d'actions et d'immobiliers dans le portefeuille, la volatilité du portefeuille augmente.

Or en simulation stochastique, avec les garanties planchers et les participations aux bénéfices, l'impact des scénarios sur le calcul de la NAV stochastique n'est pas le même. Les scénarios générant une NAV négative ont un impact plus important que ceux offrant une NAV positive. En effet, la concavité de la fonction reliant la NAV avec le profit ou la perte financière explique que les mauvais résultats financiers soient plus impactants sur la NAV en moyenne que les scénarios impactant positivement la NAV. L'asymétrie est d'autant plus importante que le portefeuille est volatil. Donc, pour le même rendement, il est préférable de choisir des obligations qui présentent une moindre volatilité.

3.5.3 Distribution des portefeuilles

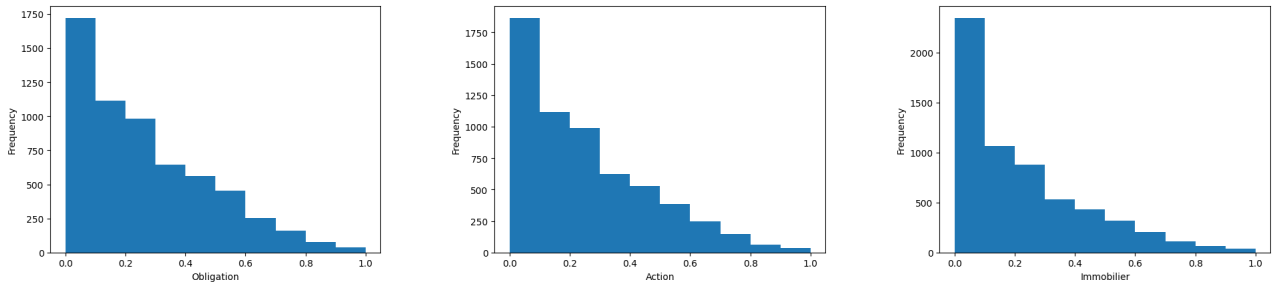

FIGURE 37 – Distribution des portefeuilles étudiées

La distribution des portefeuilles (figure 37) illustre un manque de données pour les portefeuilles extrêmes. Par conséquent, plutôt que de se limiter à une mesure globale ou à un entraînement global du modèle, il sera envisageable de mettre en place des entraînements ciblés et des mesures d'erreur plus localisées pour enrichir la compréhension du modèle. Cette distribution particulière s'explique par la contrainte suivante :

$$\text{Obligation} + \text{Action} + \text{Immobilier} \leq 1$$

3.6 Analyses des modèles

L'objectif principal est de déterminer la configuration du portefeuille cible optimale qui maximise le ratio $\frac{\text{NAV}_{\text{Central}}}{\text{SCR}_{\text{marché}}}$ en se basant sur les prévisions des différents modèles. L'étude compare les performances des modèles en terme d'erreur moyenne absolue (MAE) relative, tant globalement que par sous-modules de risque, afin d'identifier la meilleure approche pour l'allocation d'actifs.

Le choix du modèle final s'appuie sur une analyse approfondie des erreurs locales et de l'importance des variables explicatives, notamment via l'utilisation des valeurs de Shapley. De plus, la recherche du portefeuille optimal est affinée à l'aide de l'algorithme d'optimisation par essaim particulaire (PSO), permettant d'explorer de manière exhaustive l'espace des solutions possibles.

Cette étude met en lumière l'importance de combiner plusieurs approches de modélisation pour capturer les complexités des interactions entre les actifs d'un portefeuille, tout en assurant une robustesse face aux différents scénarios de marché. Le modèle final proposé, une combinaison de XGBoost et de régressions à noyaux avec pénalisation Ridge, offre une prédiction précise de la NAV, tout en maximisant l'efficience du portefeuille en termes de ratio $\frac{\text{NAV}_{\text{Central}}}{\text{SCR}_{\text{marché}}}$ à l'aide du PSO.

3.6.1 XGBoost (ensemble des données)

Ce modèle a été entrainé sur l'ensemble de la base de données, en ajoutant en entrée les classifieurs pour les différents sous-modules de risques.

Hyper-paramètres Les hyperparamètres ont été obtenus en optimisant à l'aide la librairie Optuna.

Paramètres	**Valeur**
n_estimators	57272
learning_rate	0.01
max_depth	4
min_child_weight	3
subsample	0.74
colsample_bytree	0.88
lambda	0.0005
alpha	0.0026
objective	reg :squarederror
early_stopping_rounds	50

TABLE 15 – Paramètre du modèle XGBoost entrainé sur la totalité des données

Erreurs L'erreur est disparate entre les différents modules du SCR marché. L'erreur relative moyenne est supérieure à 7% pour le sous-module risque de taux haut, alors que cette dernière est à 1% pour le module de taux bas.

Sous-module	**Relative MAE (%)**
Central	0.87
Taux_Up	7.20
Taux_Down	0.22
Action	0.37
Immobilier	0.86
Spread	1.78
Moyenne	1.89

TABLE 16 – Erreur du modèle entrainé sur la totalité des données

Erreurs locales L'erreur locale est contrôlée et reste faible sur l'ensemble du périmètre, y compris sur les potentielles zones d'intérêts.

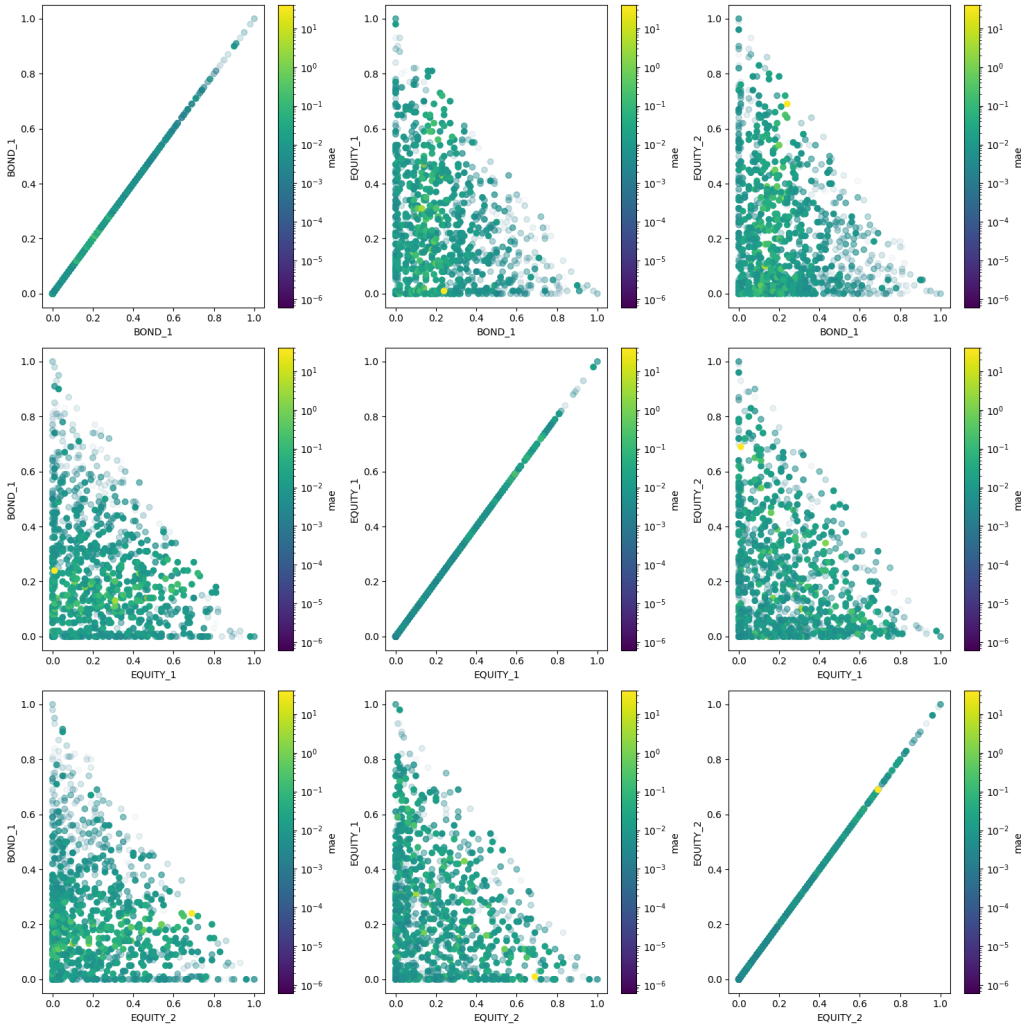

FIGURE 38 – Local MAE sur le modèle XGBoost entrainé sur la totalité des données

Features importances Le modèle semble avoir appris en donnant beaucoup de poids au classifieur "sub_module_1". Comme vu précédemment, la corrélation de ce classifieur avec la variable cible est importante, ainsi cette importance pour ce classifieur n'est pas aberrante du point de vue de l'apprentissage.
En revanche, le fait que le modèle accorde plus d'importance au classifieur "sub_module_2" par rapport à la part en obligation du portefeuille est plus surprenant. Cette sur-pondération provient du fait que ces deux classifieurs constituent les deux sous-modules les plus extrêmes pour la NAV.
La sous-pondération de la part en actions et en immobilier est cohérente vis-à-vis des données (figure 36).
Les valeurs de Shap sont en accord avec les explications proposées sur l'évolution de la NAV et l'importance des features. Les "sub_module_1" et "sub_module_2" ont des valeurs de Shap qui évoluent en opposition de phase. La part en obligations est la feature qui guide principalement la prédiction du modèle.

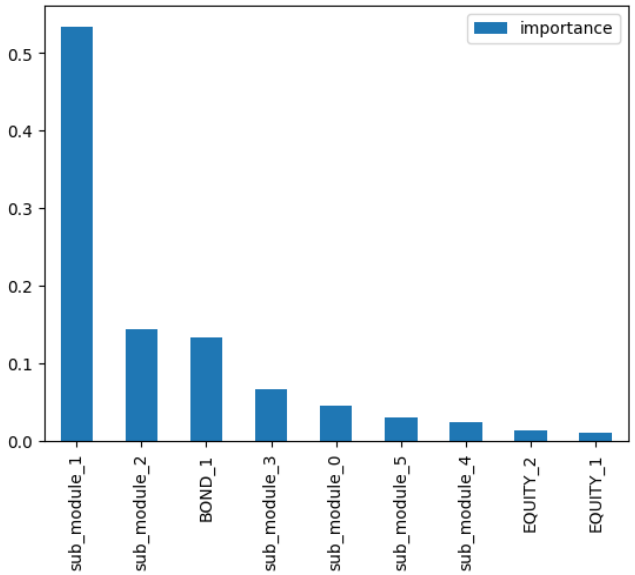

FIGURE 39 – Importance des inputs pour le modèle XGBoost entrainé sur la totalité des données

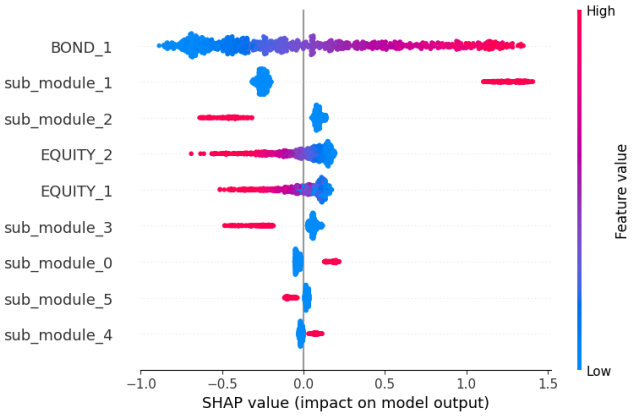

FIGURE 40 – Valeur de Shap du modèle XGBoost entrainé sur la totalité des données

Portefeuille proposé Le portefeuille maximisant la fonction objectif $\frac{\text{NAV}_{\text{Central}}}{\text{SCR}_{\text{marché}}}$ est le suivant :

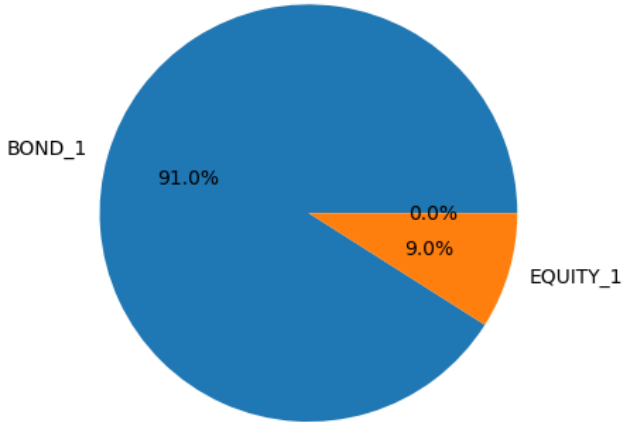

FIGURE 41 – Portefeuille optimal pour le modèle XGBoost entrainé sur la totalité des données

3.6.2 XGBoost par sous module

Hyper-paramètres Chaque modèle possède des hyperparamètres qui lui sont propres pour obtenir les meilleures performances avec son univers des possibles. Cependant, ces hyperparamètres doivent être challengés. À titre d'exemple, la profondeur du modèle "Taux bas" n'est pas forcément justifiée à la vue de ses performances par rapport au modèle général. Le concept "Wisdom of the crown" pourrait être mis en place pour diminuer davantage l'erreur. L'idée est de lancer plusieurs modèles avec des hyperparamètres parmi ceux du tableau 17 et de sélectionner les meilleurs modèles, ou bien de pondérer suivant leur score.

	Central	**Taux haut**	**Taux Down**	**Action**	**Immobilier**	**Spread**
n_estimators	9780	8494	1267	13753	186252	137143
learning_rate	0.0247	0.1062	0.1592	0.0661	0.0068	0.0455
max_depth	3	2	6	2	3	3
min_child_weight	1	1	3	1	2	3
subsample	0.7285	0.6296	0.7733	0.9360	0.6572	0.8219
colsample_bytree	0.7661	0.7333	0.5833	0.8054	0.9734	0.8324
lambda	2.90×10^{-7}	5.24×10^{-6}	2.29×10^{-5}	1.98×10^{-7}	6.04×10^{-7}	2.85×10^{-8}
alpha	0.0273	5.30×10^{-8}	9.43×10^{-8}	0.0006	1.24×10^{-8}	0.00098

TABLE 17 – Hyperparamètres de chaque XGBoost par sous module

Erreurs En subdivisant l'apprentissage du modèle par sous-modules de risque, l'apprentissage permet de diminuer l'erreur de certains modules. En effet, un entrainement sur l'ensemble de la base d'apprentissage permet au modèle d'apprendre des relations plus complexes entre les features et la variable cible.

Sous-module	**Relative MAE (%)**
Central	1.43
Taux_Up	4.39
Taux_Down	0.52
Action	0.29
Immobilier	0.43
Spread	0.44
Moyenne	1.25

TABLE 18 – Erreur du modèle XGBoost entrainé par sous module

Erreurs locales L'erreur locale reste contrôlée, il n'y a pas de zone avec des pics d'erreur très élevés.

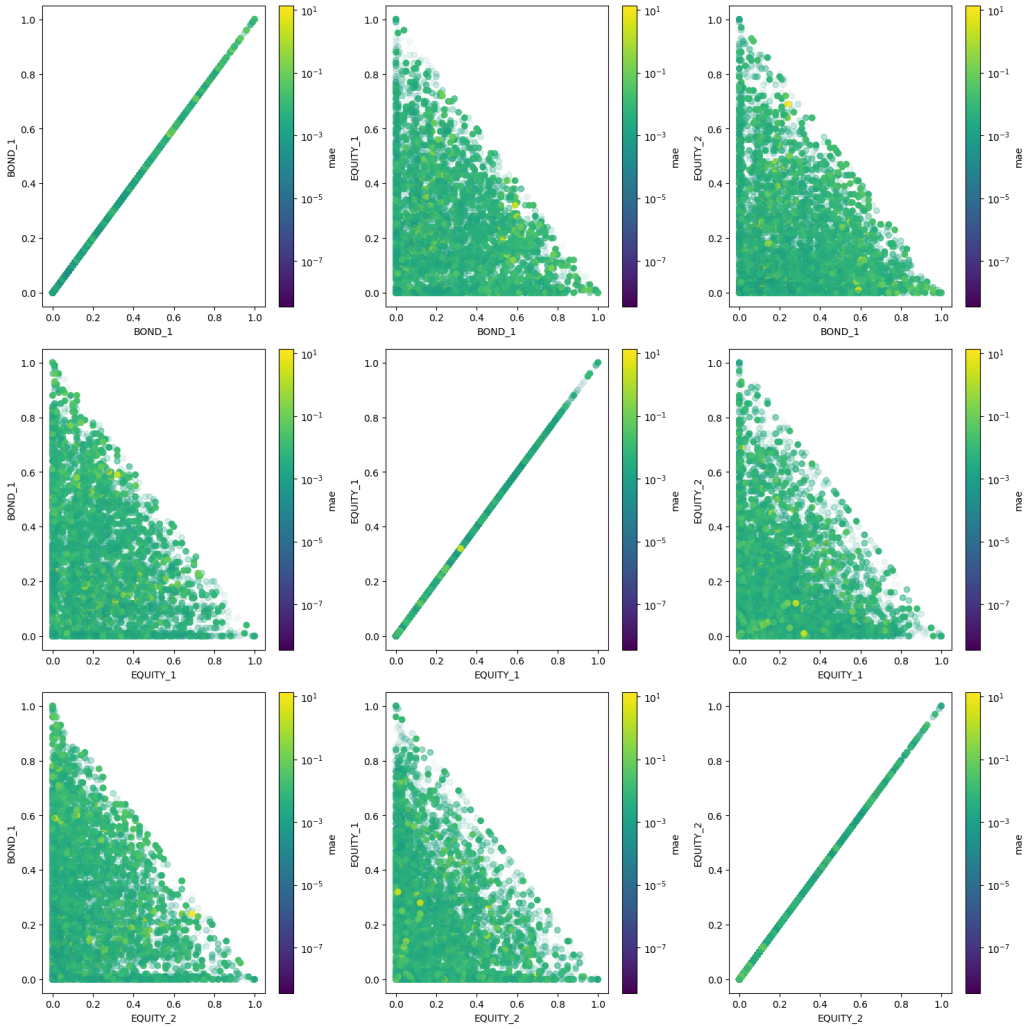

FIGURE 42 – Local MAE sur le modèle XGBoost entrainé par sous-module de risque

Features importances Comme il était prévu, les modèles donnent une importance prépondérante à la part en obligation du portefeuille. Comme attendu sur la figure 36, les valeurs de Shap du sous-module risque action témoignent de la forme d'arche caractéristique de l'évolution de la NAV en fonction de la part action. Sur la figure 44, la forme d'arche se caractérise par un repliement des valeurs de Shap sur le sous module action pour "EQUITY_1".

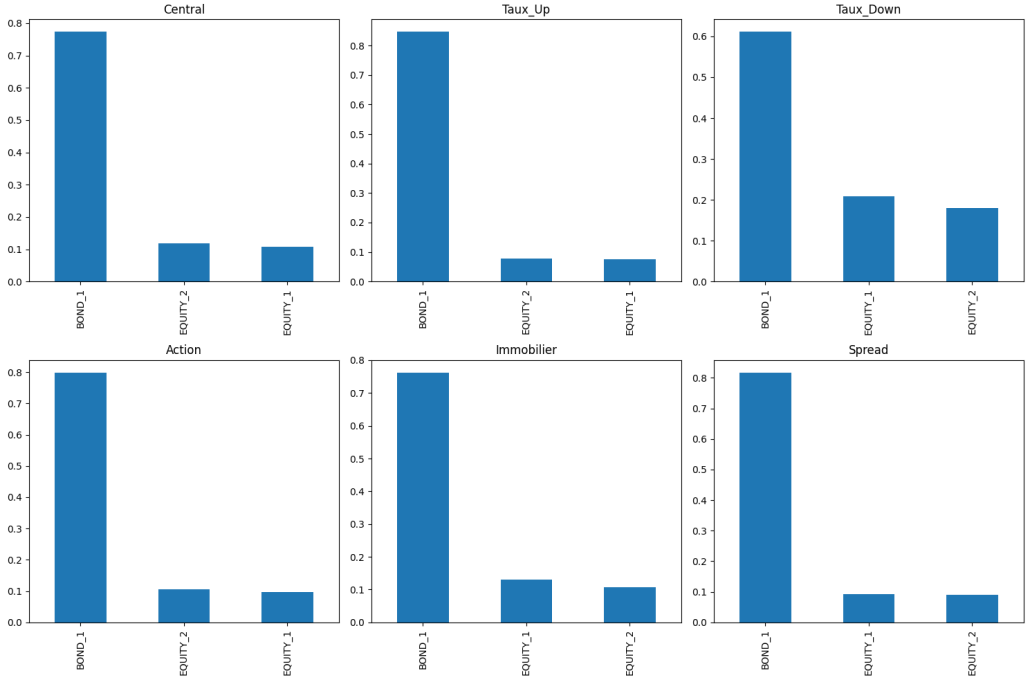

FIGURE 43 – XGBoost feature importance par sous module de risque

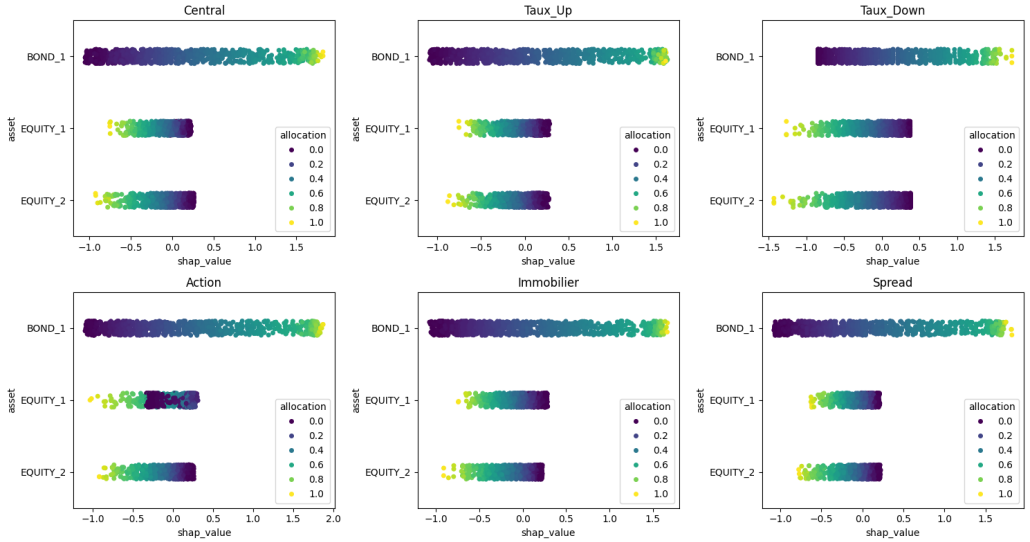

FIGURE 44 – Valeurs de Shap des modèles XGBoost entrainé sur les sous-modules

Portefeuille proposé Le portefeuille maximisant la fonction objectif $\frac{\text{NAV}_{\text{Central}}}{\text{SCR}_{\text{marché}}}$ est le suivant :

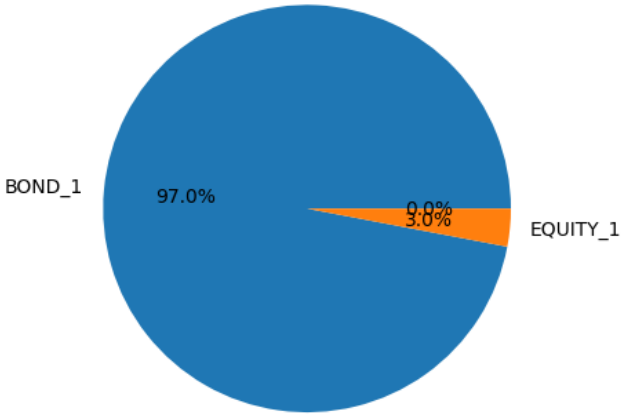

FIGURE 45 – Portefeuille optimal pour le modèle XGBoost entrainé par sous module de risque

3.6.3 Dense neural network (DNN)

Hyperparamètres La taille des modèles varie d'un sous-module à l'autre. En revanche, les modèles ont tous en commun la fonction d'activation des couches cachées et celle de la couche de sortie. En effet, avoir une fonction de sortie linéaire est adaptée avec le problème traité.

	Central	**Taux Haut**	**Taux bas**	**Action**	**Immobilier**	**Spread**
n_layers	3	2	3	3	2	4
layers_size	[192,144,80]	[432,352]	[224,208,144]	[240,208,128]	[496,464]	[464, 448, 160,96]
dense_activation	relu	relu	relu	relu	relu	relu
output_activation	linear	linear	linear	linear	linear	linear

TABLE 19 – Hyperparamètres de chaque DNN par sous module

Erreurs L'erreur obtenue varie selon les sous-modules de risque : elle est très faible pour le sous-module "Taux bas", alors qu'elle est très élevée pour le module "Spread" en comparaison avec les différents modèles de XGBoost.

Sous-module	**Relative MAE (%)**
Central	1.78
Taux_Up	3.99
Taux_Down	0.23
Action	1.21
Immobilier	6.25
Spread	3.45
Moyenne	2.82

TABLE 20 – Erreur du modèle DNN entrainé par sous-module de risque

Erreurs locales Contrairement aux modèles XGBoost, l'erreur des modèles obtenus n'est pas homogène. Des zones à forte erreur sont présentes, mais elles n'ont pas un impact significatif dans le cadre de la fonction objective. En effet, les modèles précédents et les données extraites de Prophet permettent de délimiter un espace en dehors de ces zones à forte erreur. Il est notable que ces zones de forte erreur correspondent aux espaces présentant le plus de bruit ou de ruptures. (36)

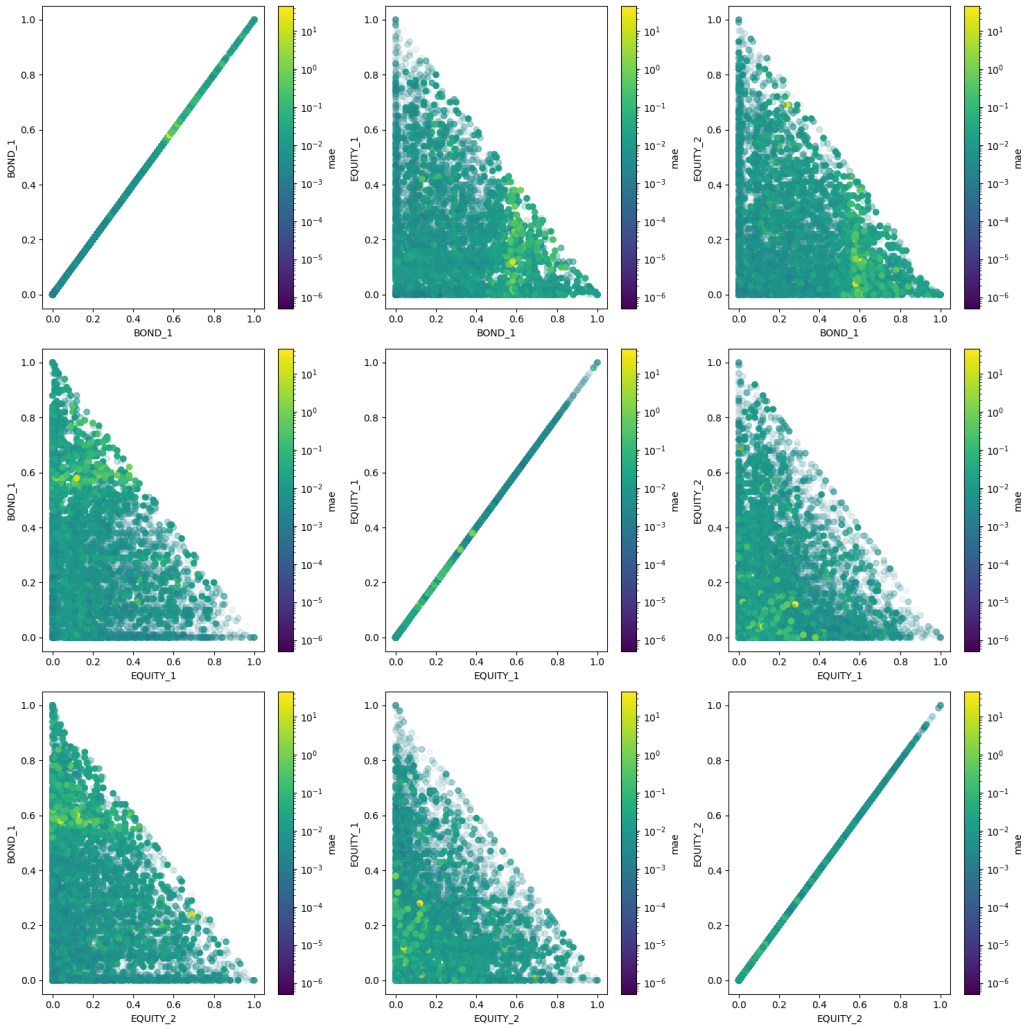

FIGURE 46 – Local MAE sur le modèle DNN entrainé par sous-module de risque

Features importance Les valeurs de Shap des différents modèles illustrent la tendance décrite par la figure 36. Pour le sous-module risque action, une non-linéarité est présente sur le diagramme. En effet, en comparant la courbe NAV en fonction de la part action du portefeuille cible, cette dernière adopte la forme d'une arche, ce qui explique le repliement observé sur le graphique 47.

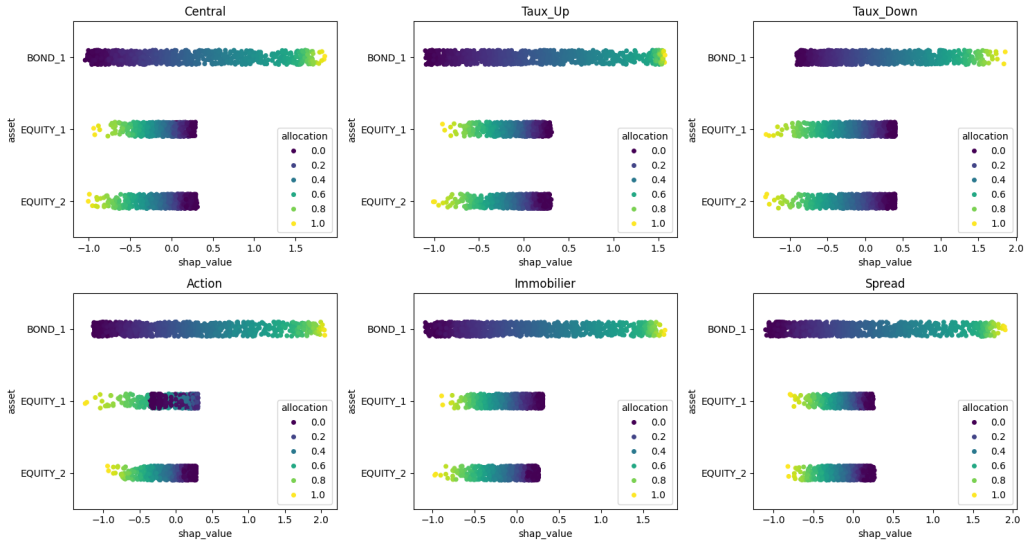

FIGURE 47 – Valeur de Shap des DNNs entrainés par sous-module de risque

Portefeuille proposé Le portefeuille maximisant la fonction objectif $\frac{\text{NAV}_{\text{Central}}}{\text{SCR}_{\text{marché}}}$ est le suivant :

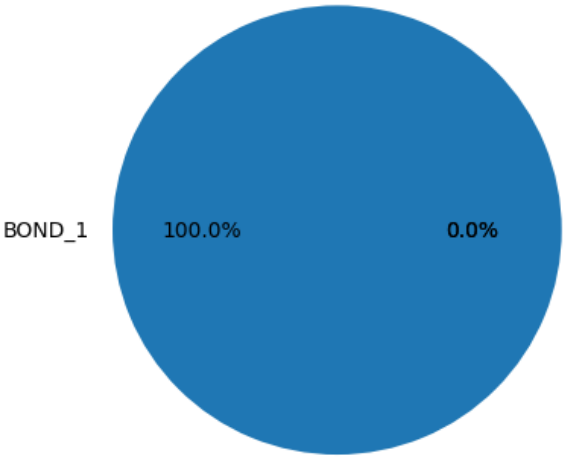

FIGURE 48 – Portefeuille optimal pour le modèle DNN entrainé par sous-module de risque

3.6.4 Kernel Ridge régression

Paramètres Les paramètres des modèles sont relativement stables, comme attendu pour un modèle de type régression. Tous les modèles ont pour noyau commun un noyau gaussien (rbf).

	Central	**Taux haut**	**Taux Down**	**Action**	**Immobilier**	**Spread**
alpha	3.2×10^{-4}	1.0×10^{-4}	1.4×10^{-7}	9.7×10^{-4}	7.0×10^{-5}	2.0×10^{-4}
gamma	18.3	23.2	4.57	37	20	21
kernel	rbf	rbf	rbf	rbf	rbf	rbf

TABLE 21 – Hyperparamètres de chaque Kernel Ridge régression par sous module

Erreurs Tout comme pour les modèles entraînés sur des sous-modules, certains sous-modules de risques sur-performent par rapport à d'autres. Par exemple, une sur-performance des régressions à noyaux avec pénalisation Ridge est observée pour le module Taux haut.

Sous-module	**Relative MAE (%)**
Central	1.54
Taux_Up	1.97
Taux_Down	0.18
Action	0.61
Immobilier	5.92
Spread	3.78
Moyenne	2.33

TABLE 22 – Erreur du modèle Kernel Ridge régression entrainé par sous-module de risque

Erreurs locales Tout comme pour les modèles denses, il existe des zones de sous-performance qui sont comparables. Par ailleurs, des zones de sur-performance sont également présentes. Elles se trouvent principalement sur les bords, ce qui est avantageux, car c'est précisément dans ces espaces stratégiques que se situent les portefeuilles optimaux.

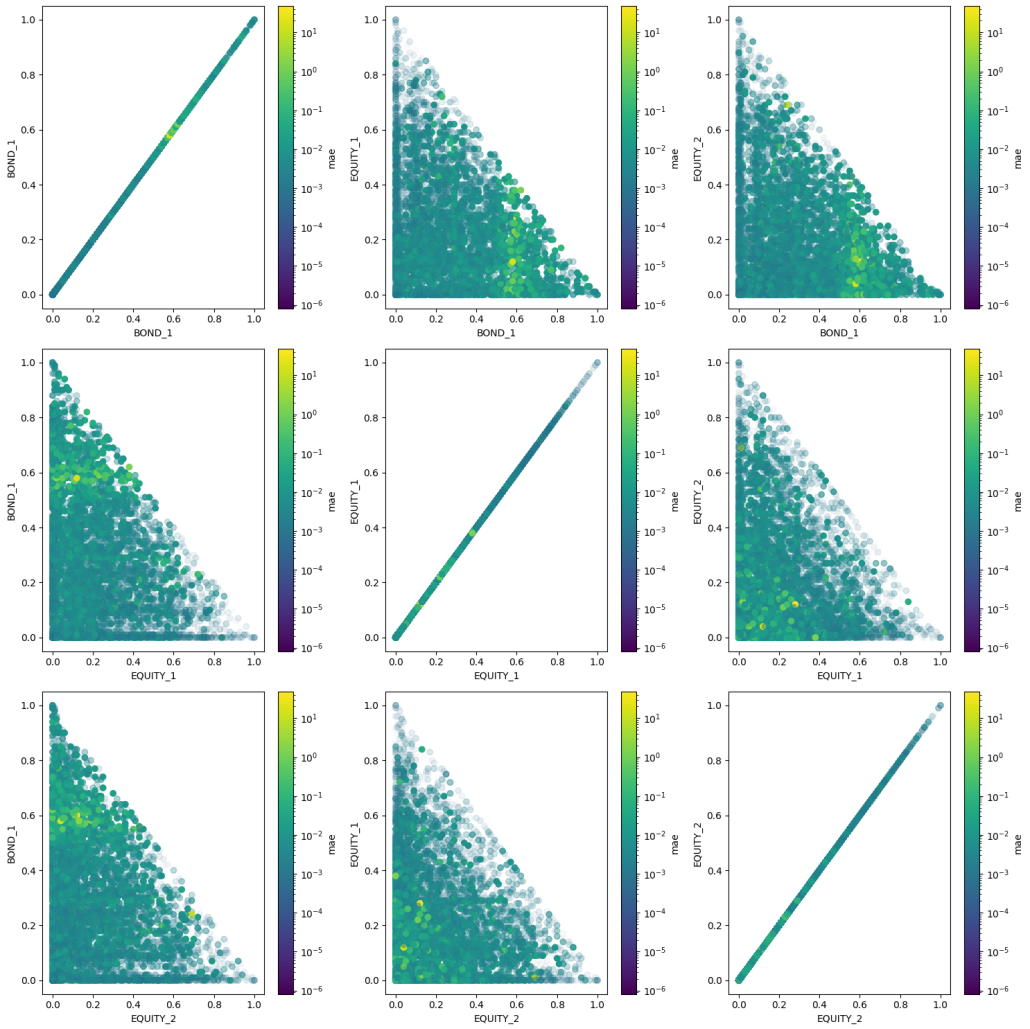

FIGURE 49 – Local MAE sur les kernel ridge régressions entrainées par sous-module de risque

Features importance La tendance exprimée par les valeurs de Shap est celle décrite par la figure 36. En revanche, les régressions à noyaux avec pénalisation de Ridge semblent capter davantage les non-linéarités que les DNNs (Dense neural network).

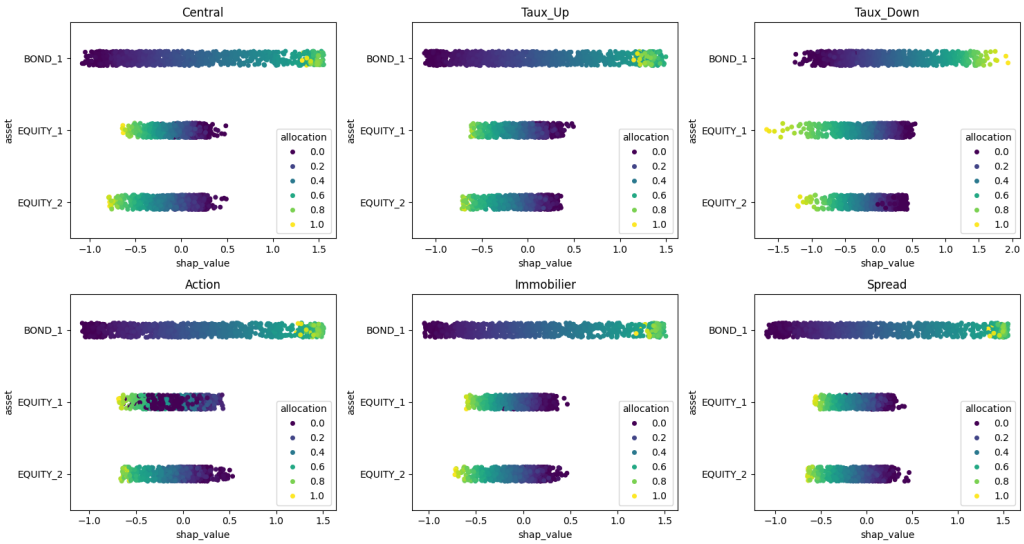

FIGURE 50 – Valeur de Shap des kernel ridge régressions entrainées par sous-module de risque

Portefeuille proposé Le portefeuille maximisant la fonction objectif $\frac{\text{NAV}_{\text{Central}}}{\text{SCR}_{\text{marché}}}$ est le suivant :

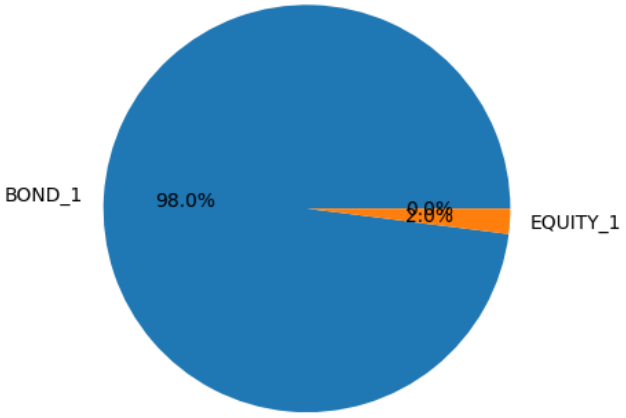

FIGURE 51 – Portefeuille optimal des kernel ridge régressions entrainées par sous-module de risque

3.6.5 Modèle final

Comparaison des modèles Dans le cadre du mémoire, le modèle désiré est celui permettant d'obtenir la meilleure connaissance de l'espace des possibles. Ainsi, pour confronter les modèles entre eux, seule l'erreur par sous-module est observée. En effet, ce procédé ne minimise pas l'erreur près du portefeuille optimal : l'erreur locale dans la zone d'intérêt aurait alors été la variable clef.

Sous-module	XGBoost	XGBoost (sous module)	DNN	Kernel Ridge régression
Central	**0.87**	1.43	1.78	1.54
Taux_Up	7.20	4.39	3.99	**1.97**
Taux_Down	0.22	0.52	0.23	**0.18**
Action	0.37	**0.29**	1.21	0.61
Immobilier	0.86	**0.43**	6.25	5.92
Spread	1.78	**0.44**	3.45	3.78
Moyenne	1.89	1.25	2.82	2.33

TABLE 23 – Relative MAE des différents modèles(%)

Le tableau 23 indique que dans le cadre de l'étude, la famille des modèles XGBoost est la plus adaptée. Les modèles de deep learning ont des performances moins bonnes. En effet, cette hiérarchie entre les modèles était à prévoir en vertu du faible nombre de données possédées. L'apprentissage profond nécessite un nombre d'échantillons plus important pour être efficace, tandis que les modèles de type arbre aléatoire et régression à noyaux possèdent de plus fortes capacités d'extrapolation et donc de meilleurs résultats avec un faible nombre d'échantillons. De plus, la régularité des NAVs dans le problème participe davantage à l'utilisation de modèles plus simples de machine learning.

Architecture D'après la table 23, le modèle retenu sera donc le suivant :

- Le modèle XGBoost entrainé sur l'ensemble de la base de donnée prédira les NAVs pour le scénario central
- Les modèles XGBoost entrainés sur des sous-modules de risque spécifiques prédiront les NAVs pour les risques Action, immobilier, spread
- Les régressions à noyaux avec pénalisation Ridge entrainées sur des sous-modules de risque spécifiques prédiront les NAVs pour les risques taux haut et taux bas

Remarque : Le choix d'une telle architecture est cohérent. Le modèle XGBoost entrainé sur tous les modules de risques est le plus capable de prédire la NAV en univers central. Les NAVs de tous les autres découlent de celle en scénario central. Ainsi, un modèle entrainé sur l'ensemble des modules possède plus d'informations et de connaissances qu'un modèle entrainé sur un sous-module spécifique. Ensuite, faire prédire les différents risques par des modèles entrainés sur le risque associé est également plus efficace, car chaque risque implique une variation de la NAV qui lui est propre.

Erreurs La MAE du modèle est donc de 0.70 % pour les données étudiées.

Portefeuille proposé Comme attendu, le portefeuille proposé est proche de celui proposé par les modèles XGBoost.

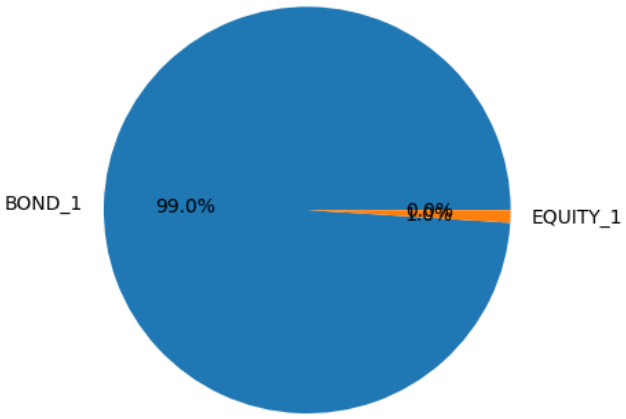

FIGURE 52 – Portefeuille optimal du modèle final

3.6.6 Recherche du portefeuille optimal

Les portefeuilles proposés ont été trouvés en utilisant un gridsearch (recherche avec une maille de l'ordre 0.1%). Pour obtenir une allocation plus précise à comparer avec le modèle ALM prophet, le Particle Swarm Optimization (PSO) sera utilisé. Cet algorithme de recherche d'optimum est utilisé pour ses capacités mêlant exploration et exploitation de l'espace.

Présentation du PSO Le Particle Swarm Optimization (PSO) est une méthode d'optimisation inspirée du comportement collectif observé dans la nature, notamment chez les essaims d'oiseaux et les bancs de poissons. Introduite par James Kennedy et Russell Eberhart en 1995, cette technique est basée sur l'idée que les individus d'un groupe peuvent trouver des solutions optimales en interagissant les uns avec les autres et en partageant des informations. Dans PSO, chaque solution possible est représentée par une "particule" dans l'espace de recherche, et ces particules évoluent en fonction de leur propre expérience ainsi que de celle de leurs voisins. En ajustant dynamiquement leur position et leur vitesse, les particules convergent vers des solutions optimales au fil du temps.

Mise à jour des particules Les équations de mise à jour des particules dans l'algorithme Particle Swarm Optimization (PSO) sont les suivantes :

Mise à jour de la vitesse

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (p_i^{best} - x_i(t)) + c_2 \cdot r_2 \cdot (g^{best} - x_i(t))$$

où :
- $v_i(t+1)$: vitesse de la particule i à l'itération $t+1$.
- $v_i(t)$: vitesse de la particule i à l'itération t.
- w : coefficient d'inertie.
- c_1 : coefficient d'accélération cognitive.
- c_2 : coefficient d'accélération sociale.
- r_1 et r_2 : variables aléatoires uniformément distribuées dans l'intervalle $[0, 1]$.
- p_i^{best} : meilleure position trouvée par la particule i.
- g^{best} : meilleure position globale trouvée par l'ensemble des particules.
- $x_i(t)$: position actuelle de la particule i à l'itération t.

Mise à jour de la position

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

où :
— $x_i(t+1)$: position de la particule i à l'itération $t+1$.
— $x_i(t)$: position de la particule i à l'itération t.
— $v_i(t+1)$: nouvelle vitesse de la particule i calculée précédemment.

Swarm design Dans le cadre du mémoire, l'essaim sera constitué de plusieurs types de particules afin d'optimiser le caractère fondé sur l'exploration et celui fondé sur l'exploitation des individus.

— **Particules standards :** elles suivent les équations classiques de PSO pour la mise à jour de la vitesse et de la position.
— **Particules à mémoire :** conservent un historique des positions et des vitesses passées, ce qui leur permet de revenir sur des zones prometteuses de l'espace de recherche. Elles exploitent davantage les connaissances acquises au fil des itérations pour affiner les solutions. [Ian14]
— **Particules exploratoires :** elles sont conçues pour maximiser l'exploration de l'espace de recherche. Elles ont des coefficients d'inertie et cognitifs élevés, ce qui les pousse à parcourir de larges zones et à éviter une convergence prématurée.
— **Particules exploitatives :** elles se concentrent sur l'exploitation des régions déjà identifiées comme prometteuses. Elles ont des coefficients sociaux élevés, les incitant à se rapprocher des meilleures solutions trouvées par l'ensemble de l'essaim, permettant ainsi une convergence rapide vers l'optimum global.
— **Particules adaptives :** elles ajustent dynamiquement leurs paramètres (comme les coefficients d'inertie, cognitifs et sociaux) en fonction de leur performance et de leur environnement. Elles peuvent ainsi passer d'une phase d'exploration à une phase d'exploitation en fonction des besoins.[Min24]
— **Particules quantiques :** elles utilisent les principes de la mécanique quantique pour explorer l'espace de recherche. Contrairement aux particules classiques, elles ne suivent pas des trajectoires déterministes, mais explorent des régions de l'espace de manière probabiliste. Cela permet de surmonter les limitations des méthodes traditionnelles et d'améliorer les capacités d'exploration, notamment dans les espaces de recherche complexes et de haute dimension.[Cle24]

Résultats du PSO La figure 53 témoigne de la convergence de l'algorithme vers le point optimal de l'espace. Le PSO permet de trouver rapidement le point d'intérêt. En moins de 30 itérations, le minimum de la fonction objective a été identifié.

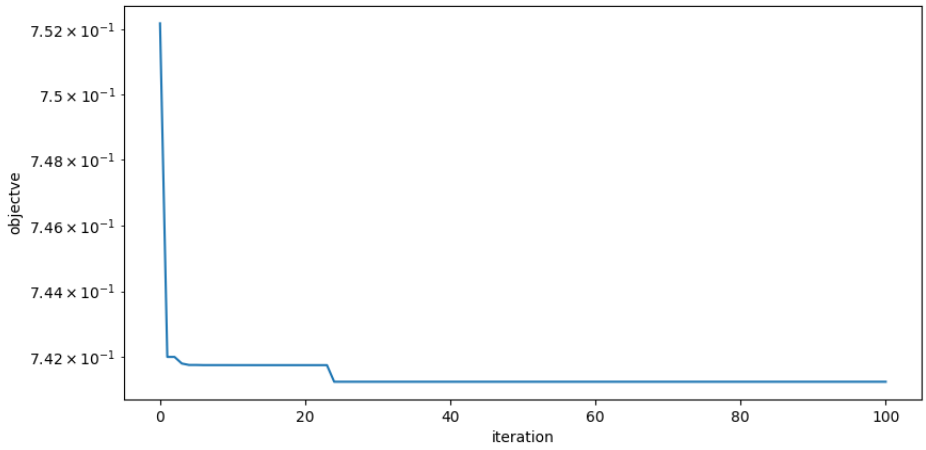

FIGURE 53 – Évolution de la fonction objective en fonction des itérations pour l'algorithme PSO

Portefeuille optimal Ainsi, selon le PSO et le modèle final, le portefeuille cible permettant d'optimiser le ratio $\frac{NAV}{SCR}$ est le suivant :

Actif	**Part**
Obligation	99%
Action	1%
Immobilier	0%
Cash	0%
Total	100%

TABLE 24 – Portefeuille optimal pour le modèle final avec recherche PSO

Ce portefeuille est cohérent, mais pourquoi avoir choisi de garder 1 % d'actions.
Afin de mieux comprendre les portefeuilles cible 100 % obligation et 99 % obligation, 1 % action sont mis en entrée du modèle ALM sous Prophet.

Portefeuille cible	100% obligation	99% obligation, 1 % action
SCR marché	10 168 145,81 €	9 820 831,07 €
NAV central	8 638 837,70 €	8 590 235,12 €
NAV action	**1 930 575,34 €**	**2 251 565,71 €**
NAV immobilier	7 858 995,53 €	7 816 458,30 €
NAV spread	7 554 935,44 €	7 506 995,66 €
NAV taux haut	13 742 092,07 €	13 746 960,28 €
NAV taux bas	5 483 186,03 €	5 422 029,05 €
$\frac{NAV}{SCR}$	85%	87%

TABLE 25 – Comparaison des portefeuilles cibles potentiels maximisant NAV/SCR (source modèle ALM)

L'ajout du 1% d'action est visible sur le SCR action, ce dernier s'en retrouve diminué sans que la NAV non choquée n'en soit grandement affectée. Ce 1% d'action permet de diminuer le SCR marché et donc d'améliorer le ratio $\frac{NAV}{SCR}$. l'impact de la part action du portefeuille cible sur le SCR marché est présenté dans la section 4.3.

4 Impact du modèle

4.1 Composition du portefeuille obligataire

La méthode d'optimisation présentée permet d'obtenir l'allocation cible idéale pour le rapport $\frac{NAV}{SCR}$ ventilée par grande catégorie d'actifs : obligation, action, immobilier. Cependant, un assureur ne choisit pas son portefeuille uniquement en fonction de ces larges classes d'actifs. En effet, un assureur doit également avoir une connaissance de son portefeuille sur plusieurs dimensions : le rating, les secteurs d'activités, la maturité, la liquidité, la matching des cash-flows, le taux de taxation, le SCR...

4.1.1 Le rating des obligations

Définition rating : Le rating d'une obligation est une évaluation attribuée par une agence de notation pour indiquer la qualité de crédit et la solvabilité de l'émetteur de l'obligation. Il reflète la capacité de l'émetteur à honorer ses obligations de paiement d'intérêts et de remboursement du principal. Les notations varient généralement de AAA, indiquant une très haute qualité et un faible risque de défaut, à D, signifiant un défaut imminent ou avéré. Ces ratings jouent un rôle crucial dans les décisions d'investissement, car ils influencent directement le rendement attendu et le niveau de risque associé à l'obligation, offrant ainsi aux investisseurs une mesure de comparaison entre différents instruments de dette.

La maille d'étude proposée pour les obligations est insuffisante pour un assureur : le portefeuille obligataire doit être ventilé par rating. La ventilation du portefeuille obligataire par rating est cruciale pour :

- Optimiser le rendement
- Gérer la volatilité
- Soutenir les stratégies d'investissement à long terme des assureurs

En diversifiant les investissements selon la qualité de crédit, les assureurs peuvent équilibrer la sécurité et la rentabilité des investissements, combinant des obligations de haute qualité pour la stabilité et des obligations de moindre qualité pour des rendements plus élevés. Cette approche permet de maximiser le rendement global tout en atténuant la volatilité, les obligations de haute qualité étant moins sensibles aux fluctuations économiques.

Adaptation du procédé : Pour répondre à cet enjeu, plusieurs alternatives sont possibles :

1. Ventiler dès le début les catégories d'actif en y incorporant le rating.
2. Faire une seconde recherche en ajoutant le rating avec la taille du portefeuille obligatoire fixée par la première recherche.

La première méthode a l'avantage d'être plus précise, mais nécessite des modèles plus complexes, car le nombre de paramètres augmente. La seconde méthode risque de proposer une précision moins bonne, en revanche les modèles restent simples. Aussi, pour conserver le plus de généralité et de rapidité sur la méthode, la seconde solution serait à privilégier.

4.1.2 Duration gap

De la même façon, il est essentiel pour un assureur de maitriser la maturité de ses actifs. Pour éviter tous risques de liquidité, la maturité des actifs doit être proche de celle du passif. Ainsi, le duration gap est un indicateur clé en gestion financière et est crucial pour les assureurs. Il mesure la différence entre la durée de sensibilité (duration) des actifs et des passifs.

La fonction objective $\frac{NAV}{SCR}$ n'est pas la plus adaptée pour optimiser la duration. De plus, la maille étudiée est trop large pour mesurer la duration précise des actifs. Ainsi, pour trouver un portefeuille à optimiser, il faut ajouter aux actifs leur maturité et les grouper par maturité afin d'optimiser le duration gap. Ainsi, la méthode proposée pourrait être réutilisée en ajoutant des classes d'actif et leur maturité, et en utilisant la fonction objective suivante, par exemple :

$$\max_x \frac{NAV}{SCR} + \exp(-|\sum_i x_i \times \text{Duration}(A_i) - \text{Duration}(P)|)$$

Où :

— A_i l'actif de classe i
— x l'allocation du portefeuille cible

4.2 Stabilité du maximum

Lorsque l'assureur définit son allocation cible, il est nécessaire d'étudier la stabilité de cette décision face à de légères perturbations. Les tests de sensibilité permettent à l'assureur d'observer l'impact de légères déviations par rapport à l'allocation cible initiale. Ces analyses permettent de déterminer si l'optimum trouvé est robuste ou si de petites variations dans les paramètres peuvent entraîner des changements significatifs dans les résultats. Une allocation cible stable doit montrer une résilience face aux fluctuations mineures des conditions de marché ou des hypothèses utilisées dans les modèles actuariels. La stabilité du maximum assure que les performances attendues de l'allocation sont fiables et stables, même en présence d'incertitudes inhérentes aux projections financières.

Cette sous-section propose d'étudier la stabilité de la Net Asset Value (NAV) en univers central en fonction de l'allocation cible envisagée. Cette analyse permet de faire un lien direct avec les normes IFRS 17 (passif) et IFRS 9 (actif). En effet, un des nombreux points de vigilance de cette norme est la stabilité de la Contractual Service Margin (CSM). Dans le cadre de cette étude, la CSM sera assimilée à la NAV : ces mesures sont des indicateurs de la richesse de l'assureur.

Introduction à IFRS 17 La norme IFRS 17, adoptée pour améliorer la comparabilité et la transparence des rapports financiers des contrats d'assurance, introduit une approche uniforme et rigoureuse dans la comptabilisation de ces contrats. Un élément central de cette norme est la CSM, qui représente les bénéfices futurs attendus des services fournis aux assurés. La CSM joue un rôle crucial dans l'alignement de la reconnaissance des revenus avec la prestation effective des services, offrant ainsi une image fidèle de la rentabilité des contrats d'assurance.

Introduction à IFRS 9 La norme IFRS 9 sur les instruments financiers, représente une évolution majeure dans la comptabilité et la gestion des risques financiers pour les entreprises. Adoptée par l'International Accounting Standards Board (IASB) et entrée en vigueur en janvier 2023 pour les assureurs concomitamment à IFRS 17, cette norme remplace IAS 39. Elle introduit des modifications significatives dans trois domaines principaux : la classification et l'évaluation des actifs financiers, la dépréciation et la comptabilité de couverture. IFRS 9 exige une approche plus prospective pour la dépréciation, reposant sur les pertes de crédit attendues, ce qui incite les entreprises à être plus proactives dans la reconnaissance et la gestion des risques de crédit.

En matière de classification, les actifs financiers doivent désormais être évalués en fonction des caractéristiques des flux de trésorerie contractuels et du modèle économique dans lequel

ils sont détenus. IFRS 9 établit trois catégories comptables pour les actifs financiers : le coût amorti, la juste valeur par le biais du résultat net (FVTPL) et la juste valeur par le biais des autres éléments du résultat global (FVOCI).

Les actifs évalués au coût amorti sont ceux détenus pour percevoir les flux de trésorerie contractuels, comme les prêts classiques. Le FVTPL s'applique aux actifs détenus à des fins de vente, avec les variations de juste valeur comptabilisées dans le résultat. Le FVOCI concerne les actifs détenus à la fois pour percevoir les flux de trésorerie et pour la vente, avec les variations de juste valeur initialement comptabilisées dans les autres éléments du résultat global. Ces catégories offrent une flexibilité accrue pour la gestion et la présentation des instruments financiers.

En pratique, les assureurs en Europe continentale ont maximisé l'utilisation de l'option juste valeur par capitaux propres (OCI), en particulier pour les actions et les obligations SPPI, les dérivés et les OPC étant quant à eux comptabilisés à la juste valeur par résultat.

La norme IFRS 17 vise à assurer que la CSM, reflète de manière précise et stable les performances financières des contrats d'assurance, même en présence de fluctuations du marché et des conditions économiques. Une allocation cible testée pour la stabilité peut ainsi atténuer la volatilité induite par les ajustements requis par IFRS 17, assurant une CSM plus stable et prévisible. Ainsi, en analysant la stabilité de la NAV en fonction de l'allocation cible, cette étude contribue à évaluer la robustesse financière de l'assureur dans un cadre régulé par IFRS 17.

Définition d'un espace à faible variation Un espace à faible variation désigne une sous-région ou un sous-ensemble de l'espace des caractéristiques (features) où les valeurs des données varient peu. Les observations ou les points de données dans cet espace sont très proches les uns des autres en termes de leurs valeurs pour certaines variables.

Mesure de la stabilité Contrairement à des tests classiques de sensibilités, cette approche permet d'obtenir une vision globale. Pour mesurer la stabilité d'une allocation, la méthode suivante sera utilisée : l'allocation la plus stable sera l'allocation au centre du plus large espace à faible variation. Le centre de cet espace est choisi pour avoir une NAV la moins sensible à l'allocation sur la zone déterminée. Par ce choix, une plus grande souplesse est offerte au portefeuille.

Méthode Le procédé se décompose en différentes étapes :

1. Calculer le gradient sur l'espace de l'allocation.
2. Mesurer la variation en sommant les valeurs absolues des composantes du gradient.
3. Filtrer pour ne garder que les points avec une faible variation aux alentours.
4. Exécuter un k-means pour obtenir des clusters
5. Sélectionner le plus grand cluster

Remarque : Un kmeans est utilisé comme algorithme de clustering, car avec une mesure L^2 il permet d'obtenir des boules, contrairement à HDBSCAN qui permet d'obtenir des clusters contigus.

Résultats : L'espace est discrétisé avec un pas spatial de $\frac{1}{300}$, soit un total de 4 500 045 points à étudier. Pour le choix du nombre de clusters, la règle du coude est utilisée. La figure 54 indique de choisir un nombre de quatre clusters. La plus grande zone à faible variation se compose de 252 021 points.

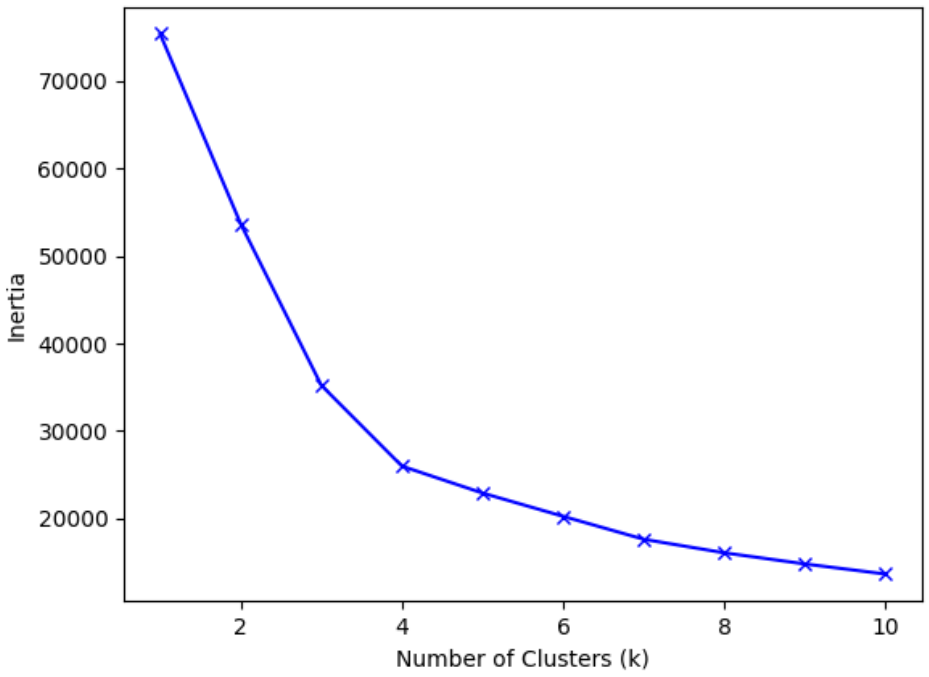

FIGURE 54 – Choix du nombre de clusters pour le kmeans avec la règle du coude

Ainsi, l'allocation la moins sensible aux variations obtenues (tableau 26) est très différente de celle optimisant le ratio $\frac{NAV}{SCR}$. L'assureur doit donc trouver un équilibre entre les différentes normes auxquelles il est soumis et ses objectifs pour obtenir une allocation répondant à l'ensemble de ces enjeux.

Actif	**Part**
Obligation	14,36 %
Action	16,30 %
Immobilier	16,65 %
Cash	46,31 %
Total	100 %

TABLE 26 – Portefeuille optimal au sens de la stabilité

Cette allocation donne une forte importance au cash et propose une répartition similaire entre les autres catégories d'actifs. Cette allocation similaire entre les actifs permet de se situer dans l'espace inférieur des allocations cibles possibles. Cet espace a l'avantage d'être le plus large possible. Ainsi, l'allocation proposée semble justifiée comme le seul critère optimisé est la taille de l'espace à faible variation.

4.3 Impact de l'aversion au risque de l'assureur

Pour élargir le sujet traité et également mieux caractériser les objectifs ou stratégies d'un assureur, la fonction objective est transformée en ajoutant un paramètre "d'aversion" au SCR κ.

$$\max_x NAV(x) - \kappa SCR(x)$$

Ce paramètre va permettre d'identifier des portefeuilles pour des assureurs avec des souhaits différents. À titre d'exemple, en prenant les cas extrêmes :

- $\kappa = 0$: l'assureur souhaite uniquement maximiser sa richesse sans regarder l'impact du SCR sur son bilan.
- $\kappa >> 1$: l'assureur souhaite obtenir le SCR le plus petit possible.

Ainsi, le paramètre κ peut être amené à changer d'un assureur à l'autre suivant le contexte économique et de marché, la stratégie à long terme ou encore le risque du portefeuille traité.

Résultats : Le point d'intérêt de cette analyse est l'étude du régime transitoire entre les deux portefeuilles extrêmes.

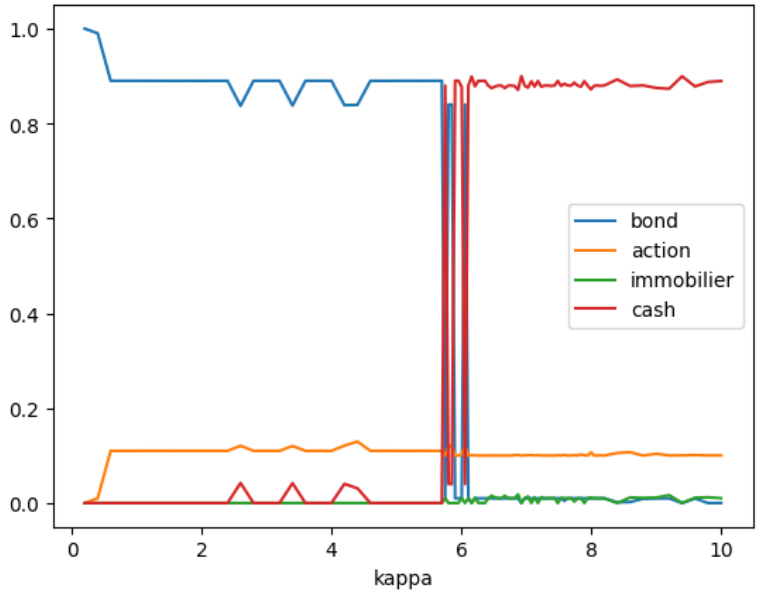

FIGURE 55 – Évolution du portefeuille cible en fonction du paramètre κ

Étude des portefeuilles extrêmes

- Le portefeuille maximisant la NAV est un portefeuille cible composé à 100 % d'obligations : une telle allocation cible est en totale cohérence avec l'univers risque neutre. En effet, pour maximiser la NAV à rendement égal, mais à volatilité moindre, les obligations sont plus intéressantes que les actions ou l'immobilier.
- Le portefeuille cible minimisant le SCR marché se compose à 90% de cash et 10 % d'action : ce portefeuille mérite une analyse plus approfondie. En effet, pour avoir un portefeuille minimisant le SCR, la solution la plus sûre et intuitive est une allocation cible en 100 % cash.

Pour valider la véracité du portefeuille minimisant le SCR marché, les extractions du modèle ALM Prophet pour une allocation cible 100 % cash et 90% cash, 10% action ont été générées.

Portefeuille cible	100% cash	90% cash, 10 % action	88% cash, 12% action
SCR marché	8 723 070,53 €	3 025 478,33 €	6 448 667,12 €
NAV central	-2 835 301,79 €	-2 857 994,36 €	-2 672 590,15 €
NAV action	**-8 566 958,47 €**	**-5 883 472,69 €**	**-6 035 030,06 €**
NAV immobilier	-3 178 570,51 €	-3 113 157,40 €	-3 001 439,30 €
NAV spread	-6 115 037,41 €	-6 143 459,75 €	-5 963 665,64 €
NAV taux haut	-3 136 990,02 €	-2 908 203,64 €	-2 679 642,63 €
NAV taux bas	-1 570 696,19 €	-1 477 730,95 €	-1 492 091,27 €

TABLE 27 – Comparaison des portefeuilles cibles potentiels minimisant le SCR (source modèle ALM)

Le modèle ALM montre qu'il est nécessaire d'inclure 10% d'actions dans le portefeuille cible pour minimiser le SCR (Solvency Capital Requirement). En effet, l'ajout d'actions permet de réduire considérablement le SCR lié au marché, en particulier celui associé au choc des actions.

Cette allocation cible est le résultat d'un équilibre entre deux dynamiques qui permettent de réduire l'écart entre les NAVs de l'univers de risque central et l'univers de risque action choquée. La première dynamique est composée de mécanismes qui forcent la vente d'actions, tandis que la seconde empêche la vente des actions. Ces différents mécanismes trouvent un point d'équilibre car leurs intensités sont différentes suivant l'allocation cible en action choisie.

Pour rappel, dans l'univers de risque central, les actions sont en plus-value, alors que dans l'univers de risque action, elles sont en moins-value. Ensuite, l'allocation cible est effectuée dès la première période de projection. Entre l'allocation cible 10% action et 0% action, la NAV central reste stable. Ainsi, l'écart entre le SCR action des allocations cibles est entièrement expliqué par les NAVs action. La différence de NAV en cas de choc des actions entre les deux portefeuilles s'explique par les éléments suivants :

— L'allocation cible, mise en place dès la première période

— Le choc action, qui entraîne une moins-value sur cette catégorie d'actifs.

Une allocation cible forçant fortement la vente des actions (exemple de l'allocation 100 % cash) implique la liquidation totale des positions en actions après le choc. Par conséquent, l'assureur réalise l'intégralité des moins-values latentes associées à ces actifs, ce qui entraîne une diminution immédiate et substantielle de la NAV. En effet, la marge financière de la première année est fortement dégradée, résultant de l'incapacité pour l'assureur à partager la perte avec les assurés.

En revanche, avec une allocation cible moins agressive sur la vente des actions, exemple 90% en cash et 10% en actions, seule une partie des positions en actions est liquidée. Ainsi, l'assureur ne réalise qu'une fraction des moins-values latentes, ce qui limite l'impact immédiat sur la NAV. En effet, l'assureur étale la réalisation de ces moins-values et est donc davantage en capacité, partiellement, de partager la perte avec les assurés au fil du temps.

Pour comprendre les raisons qui forcent la vente d'actions afin d'améliorer le SCR, l'allocation cible 12% en actions et 88% en cash est également étudiée, en comparaison avec l'allocation cible 10%. Plus particulièrement, l'impact du changement d'allocation cible sur la différence entre les NAVs en univers central et en univers action choquée est analysé.

Dans l'univers central, l'allocation cible à 12% en action permet d'atteindre une NAV plus élevée que celle à 10% en action. En effet, il n'est pas, dans ce cas, optimal pour l'assureur de réaliser trop rapidement des plus-values latentes au risque de servir aux assurés un taux plus élevé que leurs attentes. Il est plus opportun de garder une partie des richesses pour les années

suivantes, notamment afin de les utiliser dans des scénarios ou années plus difficiles (attentes des assurés plus élevées ou résultat financier dégradé).

Ensuite, dans l'univers action choquée, pour l'allocation cible à 10% en action, le taux servi la première année est plus faible que pour l'allocation cible à 12 % : l'assureur préférant dégrader le taux servi aux assurés afin de conserver partiellement sa marge financière et ne pas consommer de manière trop importante la PPE, provision pour participation aux excédents (la reprise de cette dernière étant limitée chaque année). En raison de la première valeur dégradée du taux servi, les taux servis des quelques années suivantes sont également légèrement plus faibles que pour l'allocation cible à 12% en action, du fait de la dépendance du taux cible avec le taux servi précédent. Les marges financières des années suivantes sont alors légèrement améliorées. À noter l'absence de rachat dynamique la première année car l'écart entre le taux cible et le taux servi reste en dessous du seuil de déclenchement de la loi.

En conséquence, une allocation cible d'environ 10% en actions apparaît, dans le cadre de cette étude, comme un équilibre optimal entre les différentes dynamiques présentées ci-dessus. Il est à noter que cet équilibre est fortement dépendant de la situation initiale de l'assureur et des choix pris sur la stratégie de participation aux bénéfices et de gestion de la marge financière de l'assureur.

La transition des obligations vers les actions est donc expliquée ci-dessus. Une fois le seuil d'action atteint, les obligations doivent être converties en cash pour continuer à baisser le SCR. Il est à noter que la transition d'un statut à l'autre est brutale. L'algorithme PSO ne parvient pas à capter l'ensemble de cette transition, comme illustré par la zone d'incertitude présente sur le graphique 55.

Cette étude pourra permettre à un assureur d'identifier sa situation avec son portefeuille actuel, puis en fonction de ses objectifs par rapport à son SCR et sa NAV de choisir son allocation cible optimale en conséquence. Une réserve est cependant émise quant à l'impact du point de départ sur le choix des portefeuilles cibles et les choix pris sur la stratégie de participation aux bénéfices et de gestion de la marge financière de l'assureur.

4.4 Limites et prochaines étapes

4.4.1 Transportabilité du modèle

Une limite importante de ce mémoire est l'impact de la situation initial de l'assuré du point de vue du passif comme de l'actif. Dans tout le mémoire, l'actif et le passif de l'assureur ont été fixés. Ainsi, la question de transportabilité du modèle d'un assureur à l'autre reste à étudier pour quantifier l'impact du point de départ sur le portefeuille cible optimal.

4.4.2 Étude du régime transitoire de l'allocation initiale à l'allocation cible

La prochaine étape pour optimiser l'allocation d'actif cible avec le modèle ALM est l'étude du régime transitoire entre l'allocation initiale et l'allocation cible en utilisant une fonction du type :

$$f(t) = 1 - e^{-\alpha t} \quad , \alpha > 0$$

Cette fonction servira à dimensionner l'allocation au cours du temps : plus α est grand plus la transition est rapide. Cette fonction a l'avantage de ne dépendre que d'un seul paramètre et de proposer des variations fluides. L'étude d'autres fonctions plus atypiques comme les fonctions oscillatoires ne sera cependant pas à négliger. Ensuite, une analyse de l'impact de l'allocation initiale et du paramètre α sur le portefeuille optimal est à mener pour correctement comprendre l'influence de ces différentes mécaniques sur le portefeuille cible optimal.

4.4.3 Exploration et entrainement simultanés

Faute de matériel, il a été impossible de mettre en place une approche permettant au modèle d'approximation d'explorer l'espace des allocations possibles et de s'entrainer de façon concomitante. Cette approche pourrait ainsi être très intéressante pour pallier le problème de transportabilité du modèle. En effet, en explorant et en entrainant simultanément le modèle, les temps d'entrainement et de recherche sont abaissés. De plus, cette étude permettrait de quantifier le gain de temps et la justesse apportés par cette nouvelle approche.

Conclusion

Ce mémoire a exploré une méthode innovante pour l'optimisation de l'allocation d'actifs des assureurs, en se concentrant sur le ratio $\frac{\text{NAV}_{central}}{\text{SCR}_{marche}}$. En intégrant des algorithmes de machine learning dans le processus traditionnel de gestion actif-passif (ALM), nous avons démontré qu'il est possible de réduire significativement les temps de calcul tout en améliorant la précision des prédictions et la flexibilité de l'approche.

L'utilisation d'autoencodeurs pour réduire la dimensionnalité des scénarios économiques, combinée à des techniques d'échantillonnage par maximisation de la diversité, a permis de constituer une base de données d'entraînement robuste. Cette base a ensuite été exploitée pour entraîner différents modèles de machine learning, tels que XGBoost, les réseaux de neurones denses, et les régressions à noyaux avec pénalisation Ridge. Les résultats obtenus montrent que ces modèles peuvent efficacement remplacer les méthodes de Monte Carlo traditionnelles, en fournissant des approximations précises des NAVs sous différents scénarios de risque.

L'optimisation du portefeuille d'actifs cible a été réalisée à l'aide d'un algorithme Particle Swarm Optimizer (PSO), avec des améliorations spécifiques pour équilibrer l'exploration et l'exploitation de l'espace de recherche. Cette approche a permis d'identifier des allocations d'actifs optimales qui répondent non seulement aux exigences de Solvabilité II, mais aussi aux objectifs stratégiques des assureurs, en tenant compte des contraintes et des risques spécifiques.

L'une des contributions majeures de ce mémoire réside dans la démonstration que les méthodes traditionnelles d'optimisation peuvent être enrichies par des techniques de machine learning, offrant ainsi une vision plus globale et moins locale des portefeuilles cibles. Cela permet aux assureurs de prendre des décisions mieux informées et plus adaptées aux dynamiques complexes des marchés financiers et des exigences réglementaires.

Enfin, ce travail ouvre la voie à des recherches futures, notamment sur l'étude des régimes transitoires entre les allocations initiales et cibles, et sur l'impact des différents paramètres de transition sur le portefeuille optimal. L'intégration de fonctions de transition dynamiques et l'exploration d'autres fonctions atypiques pourraient encore améliorer l'efficacité de l'optimisation de l'allocation d'actifs, offrant ainsi aux assureurs des outils encore plus puissants pour naviguer dans un environnement en constante évolution.

Référence

[Ian14] Enda Howley Ian Broderick. « Particle Swarm Optimisation with Enhanced Memory Particles ». In : *ResearchGate* (2014). doi : 10.1007/978-3-319-09952-1_24.

[IA16] Institue et Faulty of Actuaries. « Solvency II - Life Insurance ». In : (2016).

[Cle24] Maurice Clerc. « What could a Quantum PSO be ? » In : *HAL* (2024).

[Min24] Xu Yang Mingqiang Gao. « APSO-SL : An Adaptive Particle Swarm Optimization with State-Based Learning Strategy ». In : *Mdpi* (2024). doi : https://doi.org/10.3390/pr12020400.

[ACP] ACPR. *piliers solvabilité II*. url : https://acpr.banque-france.fr/controler/controle-prudentiel-des-assurances/reglementation-generale-des-assurances/.

[Ass] France Assureurs. url : https://www.franceassureurs.fr/.

[Bra] Serge Braudo. *Définition de Solvabilité/ solvable*. url : https://www.dictionnaire-juridique.com/definition/solvabilite-solvable.php.

[fin] ministère de l'économie des finances. *Quelle est la fiscalité de l'assurance-vie ?* url : https://www.economie.gouv.fr/particuliers/quelle-fiscalite-lassurance-vie.

[Leg] LegiFrance. *Code des assurances*. url : https://www.legifrance.gouv.fr/codes/.

[Pro] Prophet Professional. *Prophet Professional manual*.

[Eur] Commission Européenne. *réglement délégué solvabilité II*.

5 Annexes

5.1 Optuna

Optuna est une bibliothèque open-source en Python dédiée à l'optimisation d'hyperparamètres, particulièrement utile dans le domaine de l'apprentissage automatique. Elle se distingue par sa flexibilité et son efficacité, permettant d'optimiser une grande variété de modèles et de fonctions objectif. Optuna utilise une approche basée sur les algorithmes de recherche séquentielle et l'échantillonnage adaptatif, notamment l'algorithme Tree-structured Parzen Estimator (TPE), pour explorer de manière intelligente l'espace des hyperparamètres. Une des forces majeures d'Optuna réside dans son intégration facile avec les frameworks de machine learning populaires comme TensorFlow, PyTorch, et Scikit-learn. De plus, elle propose des fonctionnalités avancées telles que l'arrêt anticipé (pruning) pour interrompre les essais non prometteurs et économiser ainsi du temps de calcul. Optuna offre également une visualisation intuitive des résultats d'optimisation, permettant aux chercheurs et praticiens de mieux comprendre et interpréter le processus d'optimisation. En résumé, Optuna est un outil puissant et polyvalent qui facilite grandement l'optimisation des hyperparamètres, contribuant ainsi à l'amélioration des performances des modèles de machine learning.

5.2 Analyse en composante principale

L'analyse en composantes principales (ACP) est une technique statistique de réduction de la dimensionnalité souvent utilisée dans les domaines de l'analyse de données et du machine learning. L'objectif principal de l'ACP est de transformer un ensemble de variables corrélées en un ensemble de variables non corrélées, appelées composantes principales, tout en conservant le maximum d'information possible de l'ensemble de données initial. Cette méthode se décompose en plusieurs étapes :

1. **Estimation de la matrice de covariance** : Après avoir standardisé les données, pour comprendre les relations entre les variables initiales, la matrice de covariance est calculée. La matrice de covariance, notée C, est définie comme suit :

$$C = \frac{1}{n-1} Z^T Z$$

   où Z représente la matrice des données standardisées et n le nombre d'observations. Cette matrice résume comment chaque variable varie par rapport aux autres. Les éléments diagonaux de C indiquent la variance de chaque variable, tandis que les éléments non-diagonaux montrent les covariances entre les paires de variables. Une forte covariance entre deux variables suggère qu'elles varient ensemble, tandis qu'une covariance faible ou négative indique une relation moins forte ou opposée.

2. **Vecteurs et valeurs propres** : Pour identifier les composantes principales, les valeurs propres (λ) et les vecteurs propres (v) de la matrice de covariance sont calculés. Ce calcul s'effectue en résolvant l'équation caractéristique suivante :

$$Cv = \lambda v$$

   Les valeurs propres représentent la quantité de variance expliquée par chaque composante principale, tandis que les vecteurs propres indiquent la direction des nouvelles dimensions dans l'espace des données. Les valeurs propres sont ordonnées de la plus grande à la plus petite, ce qui nous permet de prioriser les composantes principales qui capturent le plus de variance. Ainsi, chaque vecteur propre associé à une valeur propre forme une nouvelle base orthogonale pour les données transformées.

3. **Calcul des composantes principales** : Une fois les valeurs propres et les vecteurs propres déterminés, les composantes principales sont calculées en projetant les données standardisées sur ces nouveaux axes. Cette projection est donnée par :

$$P = ZV$$

où V est la matrice dont les colonnes sont les vecteurs propres. Les nouvelles variables, ou composantes principales, sont des combinaisons linéaires des variables initiales. Elles sont non corrélées entre elles et ordonnées selon la quantité de variance qu'elles expliquent. En général, les premières composantes principales contiennent l'essentiel de l'information des données originales, permettant ainsi une réduction efficace de la dimensionnalité.

4. **La variance expliquée** : La proportion de la variance totale expliquée par chaque composante principale est une mesure clé pour déterminer l'importance de chaque composante. Cette proportion est calculée en divisant chaque valeur propre par la somme totale des valeurs propres :

$$\text{Variance expliquée} = \frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j}$$

où p est le nombre total de variables initiales.

5.3 Shap Value

Dans le contexte du machine learning, les modèles complexes (comme les réseaux de neurones ou les forêts aléatoires) peuvent être difficiles à interpréter. Les SHAP (SHapley Additive exPlanations) values fournissent une méthode cohérente et théoriquement fondée pour attribuer l'importance de chaque caractéristique (feature) à une prédiction individuelle. Elles sont basées sur la théorie des valeurs de Shapley, issue de la théorie des jeux.

Valeurs de Shapley en théorie des jeux Les valeurs de Shapley proviennent de la théorie des jeux et sont utilisées pour distribuer équitablement les gains ou les coûts parmi les participants d'un jeu coopératif. Elles respectent plusieurs propriétés importantes :

- **Efficacité :** La somme des valeurs de Shapley de tous les participants est égale au gain total.
- **Symétrie :** Si deux participants contribuent de la même manière, ils obtiennent la même valeur de Shapley.
- **Nullité :** Si un participant ne contribue pas du tout, sa valeur de Shapley est nulle.
- **Additivité :** Si on combine deux jeux, la valeur de Shapley pour chaque participant dans le jeu combiné est la somme des valeurs de Shapley pour chaque jeu séparé.

SHAP value en Machine Learning Les SHAP values adaptent ces concepts pour attribuer l'importance des caractéristiques dans les prédictions de modèles de machine learning. Elles fonctionnent de la manière suivante :

- **Modèle Additif :** Les SHAP values décomposent la prédiction d'un modèle en une somme d'effets attribués à chaque caractéristique, plus une valeur de base (la prédiction moyenne sur l'ensemble des données).

$$f(x) = \phi_0 + \sum_{i=1}^{M} \phi_i x_i \tag{5}$$

où $f(x)$ est la prédiction du modèle pour l'instance x, ϕ_0 est la valeur de base, et ϕ_i est la valeur SHAP pour la caractéristique i.

— **Contributions des Caractéristiques :** Chaque valeur SHAP ϕ_i représente la contribution marginale de la caractéristique i à la prédiction.
— **Calcul des SHAP values :** Le calcul des SHAP values implique de considérer toutes les combinaisons possibles de caractéristiques pour évaluer la contribution marginale de chaque caractéristique. Ce calcul peut être coûteux en termes de computation, mais des approximations et des méthodes efficaces ont été développées pour rendre ce calcul praticable pour des modèles complexes.

5.4 K-means

L'algorithme k-means est une méthode de classification non supervisée couramment utilisée en apprentissage automatique et en analyse de données. Conçu pour partitionner un ensemble de données en un nombre prédéfini de clusters, k-means assigne chaque point de données au cluster dont le centroïde est le plus proche, minimisant ainsi la variance intra-cluster. Initialement proposé par Stuart Lloyd en 1957 et popularisé par J. MacQueen en 1967, cet algorithme est apprécié pour sa simplicité conceptuelle et sa robustesse. Il est largement utilisé dans divers domaines, notamment la segmentation de marché, la reconnaissance de motifs et la compression de données. Cependant, malgré ses avantages, l'algorithme présente certaines limitations, telles que la sensibilité à l'initialisation des centroïdes et la nécessité de spécifier à l'avance le nombre de clusters, ce qui peut influer sur un résultat obtenu.

Algorithme L'algorithme k-means est une méthode de partitionnement qui vise à diviser un ensemble de données en k clusters. Son fonctionnement repose sur une procédure itérative qui suit les étapes suivantes :

1. **Initialisation** : Choisir le nombre de clusters k et initialiser les centroïdes $\mu_1, \mu_2, \ldots, \mu_k$ de manière aléatoire.
2. **Assignation** : Pour chaque point de données x_i, calculer la distance entre x_i et chaque centroïde μ_j en utilisant une mesure de distance, souvent la distance euclidienne. Assigner chaque point x_i au cluster C_j dont le centroïde μ_j est le plus proche, selon la formule :

$$C_j = \left\{ x_i : \|x_i - \mu_j\|^2 \leq \|x_i - \mu_l\|^2 \text{ pour tout } l = 1, 2, \ldots, k \right\}$$

3. **Mise à jour** : Recalculer le centroïde de chaque cluster C_j en prenant la moyenne des points qui lui sont assignés, selon la formule :

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

où $|C_j|$ est le nombre de points dans le cluster C_j.

4. **Convergence** : Répéter les étapes d'assignation et de mise à jour jusqu'à ce que les centroïdes ne changent plus significativement ou qu'un nombre maximum d'itérations soit atteint.

Ce processus itératif vise à minimiser la variance intra-cluster, assurant ainsi que chaque point de données est aussi proche que possible du centroïde de son cluster.

5.5 Gradient en dimension 3

En mathématiques, le gradient est un vecteur qui décrit la direction et le taux de variation le plus rapide d'une fonction scalaire. Pour une fonction scalaire $f : \mathbb{R}^3 \to \mathbb{R}$, définie par $f(x, y, z)$, le gradient de f en un point (x, y, z) est un vecteur noté ∇f et est défini comme suit :

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$$

où $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, et $\frac{\partial f}{\partial z}$ représentent les dérivées partielles de f par rapport à x, y, et z, respectivement.

Méthode de Calcul du Gradient

En pratique, les dérivées partielles peuvent être approximées à l'aide de différences finies. Le calcul du gradient diffère légèrement selon que l'on se trouve à l'intérieur du domaine ou à sa frontière.

— **À l'intérieur du domaine** :

$$\frac{\partial f}{\partial x} = lim_{\Delta x \to 0} \frac{f(x + \Delta x, y, z) - f(x - \Delta x, y, z)}{2\Delta x}$$

$$\frac{\partial f}{\partial y} = lim_{\Delta y \to 0} \frac{f(x, y + \Delta y, z) - f(x, y - \Delta y, z)}{2\Delta y}$$

$$\frac{\partial f}{\partial z} = lim_{\Delta z \to 0} \frac{f(x, y, z + \Delta z) - f(x, y, z - \Delta z)}{2\Delta z}$$

— **Aux frontières du domaine** :

$$\frac{\partial f}{\partial x} = lim_{\Delta x \to 0} \frac{f(x, y, z) - f(x - \Delta x, y, z)}{\Delta x}$$

$$\frac{\partial f}{\partial y} = lim_{\Delta y \to 0} \frac{f(x, y, z) - f(x, y - \Delta y, z)}{\Delta y}$$

$$\frac{\partial f}{\partial z} = lim_{\Delta z \to 0} \frac{f(x, y, z) - f(x, y, z - \Delta z)}{\Delta z}$$

Le gradient est un outil essentiel en calcul différentiel et en optimisation, permettant de comprendre comment une fonction varie dans l'espace. En dimension 3, il fournit une indication claire de la direction de variation la plus rapide de la fonction et est calculé à partir des dérivées partielles par rapport à chacune des variables.

5.6 Interaction Prophet-Python

5.6.1 Lancement des runs

Listing 5 – Classe de controle de Prophet en Python

```python
import comtypes
prophet_dll_path = ".../Prophet Professional/ProphAPI.dll"
prophet_tlb_path = ".../Prophet Professional/ProphetCallBack.tlb"
import comtypes.client
import comtypes.gen.PROPHAPILib as prophAPI
import numpy as np
import os
import pandas as pd
from tqdm import tqdm


class ProphAPISessionEventsSink:
    def __init__(self, controller):
        self.controller = controller

    def IProphProgress_NotifyPropertyChange(self, *args):
        pass

    def IProphProgress_NotifyMessage(self, Msg):
        self.controller.run_log = Msg

        if self.controller.log:
            print(f"NotifyMessage: {Msg}")

    def IProphProgress_NotifyBackupProgress(self, *args):
        pass


class ProphetController:
    def __init__(self, username='amerlin',
            log = True,var_name:str = r"code\utils\var_name.csv",
            start_period=2022, end_period = 2062,
            end_period_asset_alloc = 2072) -> None:
        self.username = username
        self.session = None
        self.event_sink = None
        self.events_connection = None
        self.reader = None
        self.log = log
        self.connection_open = False
        self.active_worspaces = []
        self.run_log = ''
        self.start_period = start_period
        self.end_period = end_period
        self.end_period_asset_alloc = end_period_asset_alloc
        self.var_name_df = self.load_var(var_name)

    def __del__(self):
        self.reader = None
        # for i in range(len(self.active_worspaces)):
        #     self.close_workspace(len(self.active_worspaces)-i-1)
        # self.close_session()
        self.session = None

    def load_var(self, var_name):
```

```
56
57
58          if type(var_name)==str:
59              var_name_df = pd.read_csv(var_name)
60          else:
61              var_label = []
62              var_period = []
63
64              for el in var_name:
65                  if el["period"] == "end":
66                      var_period.append(self.end_period)
67                      var_label.append(el["label"])
68                  elif el["period"] == "start":
69                      var_period.append(self.start_period)
70                      var_label.append(el["label"])
71                  elif el["period"] == "all":
72                      for year in range(self.start_period, self.end_period+1):
73                          var_period.append(year)
74                          var_label.append(el["label"])
75                  else:
76                      var_period.append(el["period"])
77                      var_label.append(el["label"])
78
79              var_name_df = pd.DataFrame({"var_name" : var_label, "period": var_period})
80              var_name_df["period"] = var_name_df["period"].astype(str)
81          return var_name_df
82
83      def open_session(self):
84          if self.session is None:
85              self.session = comtypes.client.CreateObject(prophAPI.ProphAPISession)
86              self.event_sink = ProphAPISessionEventsSink(self)
87              self.events_connection = comtypes.client.GetEvents(
88                          source = self.session,
89                          sink = self.event_sink)
90
91
92          res = self.session.Logon(self.username)
93          if res == 0:
94              self.connection_open = True
95              if self.log:
96                  print("Connection...Done")
97              return True
98          else:
99              raise Exception(f"Connection failed : {res}")
100
101     def close_session(self):
102         self.events_connection = None
103         self.event_sink = None
104         if self.connection_open:
105             try:
106                 res = self.session.Logoff()
107                 if res == 0:
108                     if self.log:
109                         print("Disconnection...Done")
110                     self.connection_open = False
111                     self.session = None
112                     return True
113                 else:
114                     self.session = None
115                     print(f"Disconnection failed : {res}")
```

```
            except Exception as e:
                print(f"Disconnectin failed : {e}")

    def open_workspace(self, ws_path:str) -> int:
        """
        open a workspace or if already open return the opened workspace
        """
        for i, ws in enumerate(self.active_worspaces):
            if ws.Location == ws_path:
                if self.log:
                    print(f"ws {ws_path} already open")
                return i

        ws = comtypes.client.CreateObject(prophAPI.ProphWorkspace)
        try:
            ws.Open(ws_path)
            if self.log:
                print(f"Open ws {ws_path}... Done")

            self.active_worspaces.append(ws)
            self.ws_folder = os.path.dirname(ws_path)
            return len(self.active_worspaces)-1
        except Exception as e:
            raise Exception(f'failed open ws {ws_path} : {e}')


    def close_workspace(self, ws_index):
        ws = self.active_worspaces.pop(ws_index)
        try:
            res = ws.Close()
            if res ==0:
                if self.log:
                    print(f'ws {ws_index} close ... Done')
                return True
            else:
                print(f'ws {ws_index} close ... Failed : {res}')
                return False
        except Exception as e:
            print(f'Workspace not closed : {e}')
            return False

    def run_prophet(self, run_name, structure_name, ws_index=0):
        ws = self.active_worspaces[ws_index]
        structure = ws.Structures(structure_name)
        run_setting = ws.RunSettings(run_name)

        res = run_setting.Run(structure,prophAPI.ProphRTComplete,0, self.session)

        structure = None
        run_setting = None
        if self.run_log == "Job complete" and res==0:
            self.run_log = ''
            return "success"
        else:
            print(self.run_log)
            self.run_log = ''
            return "failed"

    def read_result(self):
        if self.reader is None:
```

```
176             self.reader = comtypes.client.CreateObject(prophAPI.ProphResultsReader)
177
178     def read_result_projection(self,
179                                run_number:str,
180                                product:str,
181                                sub_product:str,
182                                varname:str,
183                                period:str,
184                                ws_path:str = None):
185         # projection = deterministe dans le reporting deloitte
186         self.read_result()
187         if ws_path is not None:
188             self.ws_folder = os.path.dirname(ws_path)
189         try:
190             value = self.reader.ProjResult(self.ws_folder,
191                                            run_number,
192                                            product,
193                                            sub_product,
194                                            varname,
195                                            str(period),
196                                            0)
197             return value
198         except Exception as e:
199             print(e)
200             return None
201
202     def read_result_stochastique(self,
203                                run_number:str,
204                                product:str,
205                                sub_product:str,
206                                varname:str,
207                                period:str,
208                                summary:str="MEAN_VALUE",
209                                ws_path:str = None):
210         # projection = deterministe dans le reporting deloitte
211         self.read_result()
212         if ws_path is not None:
213             self.ws_folder = os.path.dirname(ws_path)
214         try:
215             value = self.reader.StochasticSummary(self.ws_folder,
216                                                   run_number,
217                                                   product,
218                                                   sub_product,
219                                                   varname,
220                                                   period,
221                                                   summary)
222             return value
223         except:
224             return None
225
226     def read_result_simulation(self,
227                                run_number:str,
228                                product:str,
229                                sub_product:str,
230                                varname:str,
231                                period:str,
232                                simulation:int,
233                                ws_path:str = None):
234         # projection = deterministe dans le reporting deloitte
235         self.read_result()
```

```
        if ws_path is not None:
            self.ws_folder = os.path.dirname(ws_path)
        try:
            value = self.reader.StochasticResult(self.ws_folder,
                                                 run_number,
                                                 product,
                                                 sub_product,
                                                 varname,
                                                 period,
                                                 simulation)
            return value
        except Exception as e:
            print(e)
            return None

    def get_variable_value_prophet(self,
                                   run_number:str,
                                   product_name:str,
                                   product_code:str,
                                   variable:str,
                                   period:str,
                                   mode:str,
                                   id_simu=None,
                                   ws_path=None):
        """Retrieve a variable value from Prophet."""
        if mode not in ["det", "sto", "simu"]:
            raise ValueError("mode must be 'det' or 'sto' or 'simu'")

        if mode == "det":
            result = self.read_result_projection(run_number,
                                                 product_name,
                                                 product_code,
                                                 variable,
                                                 period,
                                                 ws_path= ws_path)
        elif mode == "sto":
            result = self.read_result_stochastique(run_number,
                                                 product_name,
                                                 product_code,
                                                 variable,
                                                 period,
                                                 ws_path=ws_path)
        else:
            result = self.read_result_simulation(run_number,
                                                 product_name,
                                                 product_code,
                                                 variable,
                                                 period,
                                                 simulation=id_simu,
                                                 ws_path=ws_path)


        return None if result == "Nan" else result

    def retrieve_result(self,
                    run_number:str,
                    product_name:str,
                    product_code:str,
                    id_simu:int = None,
                    mode:str = "det"):
```

```
296         """Retrieve results from Prophet for specified periods and variables."""
297         result = []
298         with tqdm(total=len(self.var_name_df), leave= False) as pbar:
299             if "period" in self.var_name_df.columns:
300                 for _, row in self.var_name_df.iterrows():
301                     value = self.get_variable_value_prophet(
302                                             run_number=run_number,
303                                             product_name=product_name,
304                                             product_code=product_code,
305                                             variable=row["var_name"],
306                                             period=row["period"],
307                                             mode=mode,
308                                             id_simu=id_simu)
309                     result.append(value)
310                     pbar.update(1)
311             else:
312                 raise Exception("can't use this method")
313 
314 
315         return pd.Series(result)
316 
317     def generate_asset_mix_table(self,
318                                  table_name,
319                                  asset_mix=None,
320                                  pool_2_activate = True):
321         """Generate an asset mix table for Prophet."""
322         if asset_mix is None:
323             asset_mix = {
324                 "ASSET": ["BOND", "EQUITY", "EQUITY"],
325                 "CATEGORY": [1, 1, 2],
326                 "POOL": [1, 1, 1],
327                 "ALLOCATION": [80, 10, 5]
328             }
329         if pool_2_activate:
330             asset_mix_pool_2 = {
331                 "ASSET": ["BOND", "EQUITY", "EQUITY"],
332                 "CATEGORY": [1, 1, 2],
333                 "POOL": [2, 2, 2],
334                 "ALLOCATION": [0, 0, 0]
335                 }
336 
337         asset_mix = {
338             key : asset_mix[key] + asset_mix_pool_2[key] for key in asset_mix.keys()
339             }
340 
341         content = "51\r\n!4,ASSET,CATEGORY,POOL,"
342         time_scope = [
343             str(i) for i in range(self.start_period, self.end_period_asset_alloc+1)
344             ]
345         content += ",".join(time_scope) + "\r\n"
346 
347         for i in range(len(asset_mix["ASSET"])):
348             row = f'*, "{asset_mix["ASSET"][i]}",
349                 {asset_mix["CATEGORY"][i]},
350                 {asset_mix["POOL"][i]},'
351             alloc_asset = [
352             str(int(asset_mix["ALLOCATION"][i]*100))
353             for _ in range(self.start_period, self.end_period_asset_alloc +1)
354             ]
355             row += ",".join(alloc_asset) + "\r\n"
```

```
            content += row

        content += "\r\n"
        content = content.encode('Windows-1252')

        with open(table_name, "wb") as file:
            file.write(content)

        # print("Input generated.")

    def generate_asset_starting_point(self, pool_2_activate = True):
        """Generate an asset table starting point"""
        initial_scalar = 1.19859004
        pool_basis_input = r"...\A_EURO_ASSET_POOL_BASIS.fac"
        pool_basis_output = r"...\EURO\A_EURO_ASSET_POOL_BASIS.fac"
        scalar = 1
        with open(pool_basis_input, "r") as file:
            content = file.readlines()

        line_to_replace= {
            "CASH_SCALAR" : f"*,CASH_SCALAR,{initial_scalar},1",
            "BOND_SCALAR" : f"*,BOND_SCALAR,{initial_scalar},1",
            "EQUITY_SCALAR": f"*,EQUITY_SCALAR,{scalar},1"
        }
        modified_line = []
        for line in content:
            stripped_line = line.strip()
            find_flag = False
            for key in line_to_replace.keys():
                if key in stripped_line:
                    modified_line.append(line_to_replace[key] + '\n')
                    find_flag= True
                    break
            if not(find_flag):
                modified_line.append(line)


        with open(pool_basis_output, 'w') as file:
            file.writelines(modified_line)
        if self.log:
            print("POOL_BAISIS updated")

        # bond_input = r"...\A_EURO_ASSETS_BONDS.fac"
        # bond_ouput = r"...\A_EURO_ASSETS_BONDS.fac"
        # bond_df = pd.read_csv(bond_input)
        # col_to_scale = [
        #     "I_MV",
        #     " I_FAV ",
        #     # "I_BSV",
        #     # "COUPON_AMT",
        #     # " REDEMP_AMT ",
        #                 ]
        # for col in col_to_scale:
        #     bond_df[col] = bond_df[col]*initial_scalar
        # bond_df.to_csv(bond_ouput, index=False)
        # if self.log:
        #     print("BOND table scaled")

        equity_input = r"...\A_EURO_ASSETS_EQUITY.fac"
        equity_output = r"...\A_EURO_ASSETS_EQUITY.fac"
```

```
        equity_df = pd.read_csv(equity_input)
        col_to_scale = [
            "I_MV",
            "I_FAV"
        ]
        for col in col_to_scale:
            equity_df[col] = equity_df[col] * initial_scalar
        equity_df.to_csv(equity_output, index=False)

        # cash_input = r"...\A_EURO_ASSETS_CASH.fac"
        # cash_ouput = r"...\A_EURO_ASSETS_CASH.fac"
        # cash_df = pd.read_csv(cash_input)
        # cash_df.at[1, "NOMINAL"] = cash_df.at[1, "NOMINAL"]*initial_scalar
        # cash_df.to_csv(cash_ouput, index=False)
        # if self.log:
        #     print("cash scaled")



    def generate_xsg_scenario_table(self,
                    table_name:str,
                    scenario_list:list,
                    root_scenario_file:str):
        df = pd.read_csv(root_scenario_file)
        df = df[df["SIMULATION"].isin(scenario_list)]
        unique_simulations = df['SIMULATION'].unique()
        simulation_mapping = {
            old: new+1 for new, old in enumerate(unique_simulations)
            }

        # Apply the mapping to the SIMULATION column
        df['SIMULATION'] = df['SIMULATION'].map(simulation_mapping)
        df.to_csv(table_name, index=False)

    def retrieve_result_all_period(self,
                run_number:str,
                product_name:str,
                product_code:str,
                mode:str = "det",
                write_result = False):
        """Retrieve results from Prophet for specified periods and variables."""
        df_result = pd.DataFrame(
            columns=self.var_name_df["var_name"],
            index=range(self.start_period,
            self.end_period+1)
            )

        with tqdm(total=len(self.var_name_df), leave= False) as pbar:
            for _, row in self.var_name_df.iterrows():
                for year in range(self.start_period, self.end_period+1):
                    value = self.get_variable_value_prophet(
                            run_number=run_number,
                            product_name=product_name,
                            product_code=product_code,
                            variable=row["var_name"],
                            period=year,
                            mode=mode)
                    df_result.at[year, row["var_name"]] = value
                pbar.update(1)

```

```
        if write_result:
            output_path = f"code/extraction/result_{run_number}_{mode}.csv"
            df_result.to_csv(output_path)
        return df_result
```

5.6.2 Récupération des résultats

Listing 6 – Classe d'extraction des résultats Prophet depuis Python

```
import comtypes
import comtypes.client
import comtypes.gen.PROPHAPILib as prophAPI
import numpy as np
import os
import pandas as pd
from tqdm import tqdm
from .Functions import prophet
import multiprocessing
import pythoncom
import signal



class ProphetExtractor:
    def __init__(self,
            n_simulation:int,
            extraction_file:str,
            var_name,
            run_number:str = None,
            ws_folder:str=None,
            product_name:str="A_EURO",
            product_code:str="0",
            end_period:int=2062,
            start_period:int=2022,
            name:str=None,
            username = "amerlin",
            shared_session = False) -> None:
        self.start_period = start_period
        self.end_period = end_period
        self.var_name_df = self.load_var(var_name)
        self.n_simulation = n_simulation
        self.extraction_file = extraction_file
        self.ws_folder = ws_folder
        self.product_code = product_code
        self.product_name = product_name
        self.run_number = run_number
        self.name = name
        self.username = username
        self.shared_session = shared_session


        if not(self.shared_session):
            prophet.delete_surrogate_process()
            self.session = comtypes.client.CreateObject(prophAPI.ProphAPISession)
            res = self.session.Logon(self.username)
            if res == 0:
                print("Connection...Done")
            else:
                raise Exception(f"Connection failed : {res}")

        self.reader = comtypes.client.CreateObject(prophAPI.ProphResultsReader)
```

```
53
54      def set_ws_folder(self, ws_folder):
55          self.ws_folder = ws_folder
56      def set_name(self, name):
57          self.name = name
58      def set_run_number(self, run_number):
59          self.run_number = run_number
60      def set_extraction_file(self, extraction_file):
61          self.extraction_file = extraction_file
62
63      def load_var(self, var_name):
64          if type(var_name)==str:
65              var_name_df = pd.read_csv(var_name)
66          else:
67              var_label = []
68              var_period = []
69
70              for el in var_name:
71                  if el["period"] == "end":
72                      var_period.append(self.end_period)
73                      var_label.append(el["label"])
74                  elif el["period"] == "start":
75                      var_period.append(self.start_period)
76                      var_label.append(el["label"])
77                  elif el["period"] == "all":
78                      for year in range(self.start_period, self.end_period+1):
79                          var_period.append(year)
80                          var_label.append(el["label"])
81                  else:
82                      var_period.append(el["period"])
83                      var_label.append(el["label"])
84
85              var_name_df = pd.DataFrame({"var_name" : var_label, "period": var_period})
86              var_name_df["period"] = var_name_df["period"].astype(str)
87          return var_name_df
88
89      def read_result_stochastic(self,varname:str, period:str, simulation:int):
90          try:
91              value = self.reader.StochasticResult(self.ws_folder,
92                      self.run_number,
93                      self.product_name,
94                      self.product_code,
95                      varname,
96                      period,
97                      simulation)
98              return value
99          except Exception as e:
100             raise e
101
102     def retrieve_result(self, id_simu:int, pbar_disable=True):
103         result = []
104         with tqdm(total=len(self.var_name_df), disable=pbar_disable) as pbar:
105             for _, row in self.var_name_df.iterrows():
106                 value = self.read_result_stochastic(
107                     varname=row["var_name"],
108                     period=row["period"],
109                     simulation=id_simu
110                     )
111                 result.append(value)
112                 pbar.update(1)
```

```
113         return result
114
115     def extraction(self, pbar_disable=False):
116         columns = [
117           row["var_name"]+"_"+str(row["period"]) for _, row in self.var_name_df.iterro
118         ]
119         columns = ["SIMULATION"] + columns
120         rows = []
121         if os.path.exists(self.extraction_file):
122             last_checkpoint = pd.read_csv(self.extraction_file)["SIMULATION"].max()
123         else:
124             last_checkpoint = -1
125         for id_simu in tqdm(range(1, self.n_simulation+1), disable= pbar_disable):
126             if id_simu>last_checkpoint:
127                 row = self.retrieve_result(id_simu=id_simu)
128                 row = [id_simu] + row
129                 rows.append(row)
130                 if len(rows)>100:
131                     tmp_df = pd.DataFrame(rows, columns=columns)
132                     tmp_df.to_csv(
133                         self.extraction_file,
134                         mode="a",
135                         header= not os.path.exists(self.extraction_file),
136                         index=False
137                         )
138                     rows = []
139
140         tmp_df = pd.DataFrame(rows, columns=columns)
141         tmp_df.to_csv(
142             self.extraction_file,
143             mode="a",
144             header= not os.path.exists(self.extraction_file),
145             index=False
146             )
147         rows = []
148
149
150     def __del__(self):
151         self.reader = None
152         if not(self.shared_session):
153             prophet.delete_surrogate_process()
154
155
156 class ExtractorManager:
157     def __init__(self, setting, username:str = "amerlin") -> None:
158         self.setting = setting
159         self.nb_parallal_extraction = ExtractorManager.check_input(setting)
160         self.username = username
161
162         prophet.delete_surrogate_process()
163         self.session = comtypes.client.CreateObject(prophAPI.ProphAPISession)
164         res = self.session.Logon(self.username)
165         if res == 0:
166             print("Connection...Done")
167         else:
168             raise Exception(f"Connection failed : {res}")
169
170         self.init_extractors()
171
172     @staticmethod
```

```
    def check_input(setting):
        size = None
        for key in setting:
            if size is None:
                size = len(setting[key])
            else:
                assert len(setting[key]) == size
        return size
    def init_extractors(self):
        self.extractors = []
        for i in range(self.nb_parallal_extraction):
                self.extractors.append(
                    ProphetExtractor(
                                n_simulation= self.setting["n_simulation"][i],
                                extraction_file= self.setting["extraction_file"][i],
                                var_name= self.setting["var_name"][i],
                                ws_folder= self.setting["ws_folder"][i],
                                name =self.setting["name"][i] ,
                                run_number= self.setting["run_number"][i],
                                shared_session=True
                                )
                )

        print("Extractor are initiazed")

    @staticmethod
    def worker(task_num, extractor:ProphetExtractor):
        print(f"{task_num} start")
        pythoncom.CoInitialize()

        extractor.extraction()

        pythoncom.CoUninitialize()
        print(f"{task_num} end")

    @staticmethod
    def handle_interrupt(signum, frame):
        for p in multiprocessing.active_children():
            p.terminate()
        exit(1)

    def multi_extraction(self):
        signal.signal(signal.SIGINT, ExtractorManager.handle_interrupt)

        processes = []
        for i, extractor in enumerate(self.extractors):
            p = multiprocessing.Process(
                target=ExtractorManager.worker,
                args=(i, extractor)
            )
            processes.append(p)
            p.start()

        for p in processes:
            p.join()

    def __del__(self):
        if hasattr(self, "extractors"):
            for extractor in self.extractors:
                del extractor
```

```
            prophet.delete_surrogate_process()
```

5.7 Particle Swarm Optimizer

Listing 7 – Classe pour l'essaim de particules

```
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
import random as rd
from .particles import  StandardParticle,
                        ExploitativeParticle,
                        ExploreativeParticle,
                        AdaptiveParticle,
                        MemoryParticle,
                        QuantumParticle

import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

class ParticleSwarmOptimizer:
    def __init__(self,
            objective_function,
            bounds,
            num_particles=30,
            max_iterations=100,
            store_positions=False):
        self.objective_function = objective_function
        self.store_position = store_positions
        self.bounds = bounds
        self.num_particles = num_particles
        self.max_iterations = max_iterations
        self.swarm = self.initialize_swarm()
        self.global_best_position = self.calculate_population_best()
        self.global_best_value = min([particle.best_value for particle in self.swarm])
        self.global_best_values = [self.global_best_value]

    def initialize_swarm(self):
        swarm = []
        particle_classes = [
            (StandardParticle, {
                'inertia': 0.5,
                'cognitive': 1.5,
                'social': 1.5,
                'store':self.store_position}),
            (ExploreativeParticle, {
                'inertia': 0.9,
                'cognitive': 2.0,
                'social': 2.0,
                'store':self.store_position}),
            (ExploitativeParticle, {
                'inertia': 0.3,
                'cognitive': 1.0,
                'social': 2.5,
                'store':self.store_position}),
            (AdaptiveParticle, {
                'inertia': 0.5,
                'cognitive': 1.5,
                'social': 1.5,
                'store':self.store_position}),
            (MemoryParticle, {
```

```
                'inertia': 0.5,
                'cognitive': 1.5,
                'social': 1.5,
                'memory_size' : 5,
                'store':self.store_position}),
            (QuantumParticle, {
                'inertia': 0.5,
                'cognitive': 1.5,
                'social': 1.5,
                'store':self.store_position})
        ]
        for _ in range(self.num_particles):
            position = np.random.uniform(
                [b[0] for b in self.bounds], [b[1] for b in self.bounds]
                )
            position = self.apply_constraints(position)
            velocity = np.random.uniform(-1, 1, len(self.bounds))
            particle_class, params = rd.choice(particle_classes)
            swarm.append(particle_class(
                position,
                velocity,
                self.objective_function,
                **params)
                )
        return swarm

    def calculate_population_center(self):
        positions = np.array([particle.position for particle in self.swarm])
        return np.mean(positions, axis=0)

    def calculate_population_best(self):
        best_particle = min(self.swarm, key=lambda p: p.best_value)
        return best_particle.best_position

    def get_all_best_positions(self):
        return [particle.best_position for particle in self.swarm]

    def apply_constraints(self, position):
        # constraint: x1 + x2 + x3 < 1
        position = np.clip(
                    position,
                    [b[0] for b in self.bounds],
                    [b[1] for b in self.bounds]
                )
        if np.sum(position) > 1:
            excess = np.sum(position) - 1
            package = excess /10
            nb_package = 0
            while nb_package <10:
                i = rd.choice([j for j in range(len(position))])
                if position[i]- package > 0:
                    position[i] -= package
                    nb_package +=1

        position = np.clip(
                    position,
                    [b[0] for b in self.bounds],
                    [b[1] for b in self.bounds]
                )

```

```
        return position

    def run(self):
        for _ in tqdm(range(self.max_iterations)):
            population_center = self.calculate_population_center()
            best_positions = self.get_all_best_positions()

            for particle in self.swarm:


                if isinstance(particle, AdaptiveParticle):
                    particle.update_velocity(
                            self.global_best_position,
                            population_center,
                            best_positions,
                            self.bounds
                        )
                elif isinstance(particle, MemoryParticle):
                    particle.update_velocity(self.global_best_position)
                    particle.update_memory()
                elif isinstance(particle, QuantumParticle):
                    particle.update_velocity(self.global_best_position, self.bounds)
                    particle.apply_quantum_rotation()
                elif isinstance(particle, ExplorativeParticle):
                    particle.update_velocity(self.global_best_position, self.bounds)
                elif isinstance(particle, StandardParticle)
                        or isinstance(particle, ExploitativeParticle):
                    particle.update_velocity(self.global_best_position)

                particle.update_position(
                    self.bounds,
                    apply_constraint=self.apply_constraints
                )

                if particle.best_value < self.global_best_value:
                    self.global_best_position = particle.best_position.copy()
                    self.global_best_value = particle.best_value

            self.global_best_values.append(self.global_best_value)

            print(self.global_best_position, self.global_best_value)

        return self.global_best_position, self.global_best_value

    def plot_history_value(self):
        plt.plot(self.global_best_values)
        plt.yscale("log")
        plt.xlabel("iteration")
        plt.ylabel("objectve")
        plt.show()

    def animate(self, interval=2000, margin = 1/10):
        fig = plt.figure(figsize=(10,6))
        xlim = (self.bounds[0][0] - margin, self.bounds[0][1]+ margin)
        ylim = (self.bounds[1][0] - margin, self.bounds[1][1]+ margin)
        ax = plt.axes(xlim=xlim, ylim=ylim)
        ax.set_xlabel("Obligation")
        ax.set_ylabel("Action")
        particle_colors = {
            StandardParticle: ('bo', 'Standard Particle'),
```

```
            ExploitativeParticle: ('ro', 'Exploitative Particle'),
            AdaptiveParticle: ('co', 'Adaptive Particle'),
            MemoryParticle: ('mo', 'Memory Particle'),
            QuantumParticle: ('yo', 'Quantum Particle')
        }

        scatter_plots = {
    ptype: ax.plot([], [], style)[0] for ptype, (style, _) in particle_colors.items()
        }


        def init():
            for plot in scatter_plots.values():
                plot.set_data([], [])
            return scatter_plots.values()

        def update(frame):
            xdata = {ptype: [] for ptype in particle_colors}
            ydata = {ptype: [] for ptype in particle_colors}
            for particle in self.swarm:
                ptype = type(particle)
                xdata[ptype].append(particle.positions[frame][0])
                ydata[ptype].append(particle.positions[frame][1])

            for ptype, plot in scatter_plots.items():
                plot.set_data(xdata[ptype], ydata[ptype])


            return scatter_plots.values()

        legend_elements = [plt.Line2D(
                            [0],
                            [0],
                            marker='o',
                            color='w',
                            markerfacecolor=style[0],
                            markersize=10,
                            label=label
                            )
                    for (style, label) in particle_colors.values()]
        ax.legend(handles=legend_elements)

        ani = FuncAnimation(
            fig,
            update,
            frames=self.max_iterations,
            init_func=init,
            blit=True,
            interval=interval
        )
        plt.show()
```

Table des figures

Liste des tableaux

Listings