**Mémoire présenté le :**

**pour l'obtention du Diplôme Universitaire d'actuariat de l'ISFA et l'admission à l'Institut des Actuaires**

Par : Philippe EAR

Titre : Estimation du SCR sous formule standard par des méthodes de machine learning

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

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

*Membres présents du jury de l'Institut des Actuaires* signature

*Membres présents du jury de l'ISFA*

*Entreprise :*

Nom : CNP Assurances

*Signature :*

*Directeur de mémoire en entreprise :*

*Nom : Tarek AOUDI*

*Signature :*

*Invité :*

*Nom :*

*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

Signature du candidat

# Remerciements

J'aimerais remercier l'ensemble des personnes qui ont pu m'accompagner durant mon expérience chez CNP Assurances et pendant la rédaction de mon mémoire sur l'estimation du SCR d'une compagnie d'assurance sous formule standard par le biais de machine learning.

Parmi toutes ces personnes, je remercie tout particulièrement Tarek AOUDI, anciennement Responsable Risques qui était mon encadrant chez CNP Assurances et maintenant CRO de CNP Luxembourg, pour ses précieux conseils qu'il a pu me donner. Ses connaissances et son expérience m'ont été d'une grande aide pour l'accomplissement de ce mémoire.

Jérémy ROBERT, Responsable Capital et Solvabilité de CNP Assurances, pour sa bienveillance et ses conseils, et sans qui je n'aurais peut-être pas eu cette opportunité chez CNP Assurances.

Misa RATIARAY ainsi que toute l'équipe RS2 pour m'avoir accueilli parmi eux et contribué à ma bonne insertion dans l'entreprise durant cette période, de même pour les multiples stagiaires et alternant avec qui j'ai pu partager mes repas et journées.

Et je remercie également Diana DOROBANTU, ma tutrice académique, ainsi que l'ensemble du corps enseignant de l'ISFA Lyon du M2 Actuariat. Cette formation de qualité m'a permis de comprendre les enjeux et d'avoir les outils nécessaires pour répondre aux problématiques du métier et de l'assurance en général.

## Résumé

La réglementation Solvabilité 2 nécessite d'avoir un processus officiel pour la production du capital de solvabilité requis et du SCR. Ce processus peut être long et lourd à mettre en place et à utiliser, notamment lorsque la compagnie d'assurance passe par la formule standard, ce qui la rend moins réactive dans le cadre d'évènements soudains nécessitant une prise de décision rapide de sa direction. C'est dans ce contexte que des méthodes alternatives de production de SCR sont développées au sein des entreprises. Ces méthodes sont souvent approximatives, mais beaucoup plus performantes et sont utilisées comme proxy dans ces situations rares qui nécessitent une évaluation rapide du coût en SCR d'une décision. Parmi ces méthodes, les techniques utilisant les modèles de machine learning sont souvent plébiscités pour leurs performances et leurs simplicités d'implémentation.

Le but de ce mémoire de créer un premier outil permettant de faire des estimations de SCR rapide dans le cas d'une entreprise sous formule standard.

Pour cela deux grandes étapes seront nécessaires :

— Utiliser les données de haute qualité afin de créer une base de données pouvant servir de base d'entraînement et de validation.

— Entraîner et optimiser des modèles de machine learning, notamment le Random Forest et le XGBoost pour estimer le SCR dans différents scénarios.

Une étape cruciale lors de la sélection des modèles de machine learning est l'étape du paramétrage. Les deux principaux modèles étudiés disposent de nombreux hyperparamètres pouvant grandement influencer la performance de ces derniers. Une approche par grid-search permettra de déterminer des jeux de paramètres localement optimaux avec lesquels travailler.

Les résultats de ces méthodes seront finalement étudiés par le biais d'analyses de similitude par rapport à la distribution empirique, d'erreurs relatives par choc et finalement d'erreurs relatives globales.



**Mots-clés :** Solvabilité 2, formule standard , apprentissage statistique, machine learning, extreme gradient boosting, forêt aléatoire, arbre de régression, régression LASSO, recursive feature elimination, base de données, grid-search

## Abstract

Solvency 2 regulation requires an official process for producing the required solvency capital and SCR. This process can be lengthy to implement and use, especially when the insurance company uses the standard formula, which makes it less reactive in the event of sudden events requiring quick decision-making by its management. In this context, alternative methods of SCR production are developed within companies. These methods are often approximate but much more effective and are used as proxies in these rare situations that require a quick evaluation of the SCR cost of a decision. Among these methods, techniques using machine learning models are often favored for their performance and simplicity of implementation.

The purpose of this thesis is to create a first tool for quickly estimating SCR in the case of a company using the standard formula.

To do this, two main steps will be necessary :

— Use high-quality data to create a database that can serve as a training and validation base.

— Train and optimize machine learning models, including Random Forest and XGBoost, to estimate SCR in different scenarios.

A crucial step in the selection of machine learning models is the parameterization step. Both of the main models studied have many hyperparameters that can greatly influence their performance. A grid-search approach will allow for determining locally optimal sets of parameters to work with.

The results of these methods will finally be studied through similarity analyses with respect to the empirical distribution, relative errors by shock, and finally overall relative error.



Keywords : Solvency 2, standard formula, statistical learning, machine learning, extreme gradient boosting, random forest, regression tree, LASSO regression, recursive feature elimination, database, grid-search

# Synthèse

## Problématique

Avec la réglementation Solvabilité 2, plusieurs techniques ont été proposées afin de calculer le capital réglementaire pour les compagnies d'assurance, dont la formule dite standard. Cette méthode permet de ne pas avoir besoin de développer son propre modèle interne, mais est en revanche assez lourde à mettre en place et rigide dans ses processus. C'est notamment la technique qu'utilise CNP Assurances afin de produire son SCR et donc son ratio de solvabilité.

Cependant avec des marchés instables, il est nécessaire de pouvoir évaluer l'impact d'un choc issu de sensibilités de manière rapide et fiable afin d'orienter des décisions de la direction de l'entreprise. La méthode classique est trop lente pour pouvoir répondre à ces besoins et les techniques utilisées actuellement ne sont alors souvent pas beaucoup plus complexes qu'un produit en croix.

Dans ce mémoire, une nouvelle technique va être développée et testée afin d'estimer l'impact d'un choc sur les SCR de l'entreprise en utilisant des méthodes de machine learning.

## Création de la base de données

Premièrement, une base de données servant à la fois de base d'entraînement et de validation pour nos modèles de machine learning sera créée. Cette base de données utilise 40 sensibilités pour la clôture annuelle de fin d'année et contient plus de 700 000 lignes avec plus d'une centaine de variables. Afin de créer cette base de données, une multitude d'étapes ont été nécessaires, allant de la récupération des fichiers bruts, à la transformation et nettoyage des données, pour finalement obtenir une base propre contenant des variables pertinentes pour les futurs travaux.

Les variables ont pu être modifiées pour réduire le nombre de modalité, transformer une variable qualitative en une variable quantitative ou encore transformer l'unité de la variable. Ces variables sont multiples, avec des variables caractérisant le code S2 de l'actif considéré, leurs valeurs boursière ou encore quel type de choc est actuellement considéré.

| | ALLOCATION_STRATEGIQUE_Diversification | ALLOCATION_STRATEGIQUE_Taux | ... | ChocSpreadObligation | ChocSpreadTitrisation |
|---|---|---|---|---|---|
| 0 | 0.222299 | 0.733834 | ... | False | False |
| 1 | 0.222299 | 0.733834 | ... | False | False |
| 2 | 0.222299 | 0.733834 | ... | False | False |
| 3 | 0.222299 | 0.733834 | ... | False | False |
| 4 | 0.222299 | 0.733834 | ... | False | False |
| ... | ... | ... | ... | ... | ... |
| 720715 | 0.222299 | 0.733834 | ... | False | False |
| 720716 | 0.222299 | 0.733834 | ... | False | False |
| 720717 | 0.222299 | 0.733834 | ... | False | False |
| 720718 | 0.222299 | 0.733834 | ... | False | False |
| 720719 | 0.222299 | 0.733834 | ... | False | False |

720720 rows × 143 columns

Extrait de la base de données du T4 2021

### Optimisation et entraînement des modèles

A partir de cette base de données, plusieurs modèles de machine learning seront testés :

— Régression LASSO

— Random Forest

— eXtreme Gradient Boost

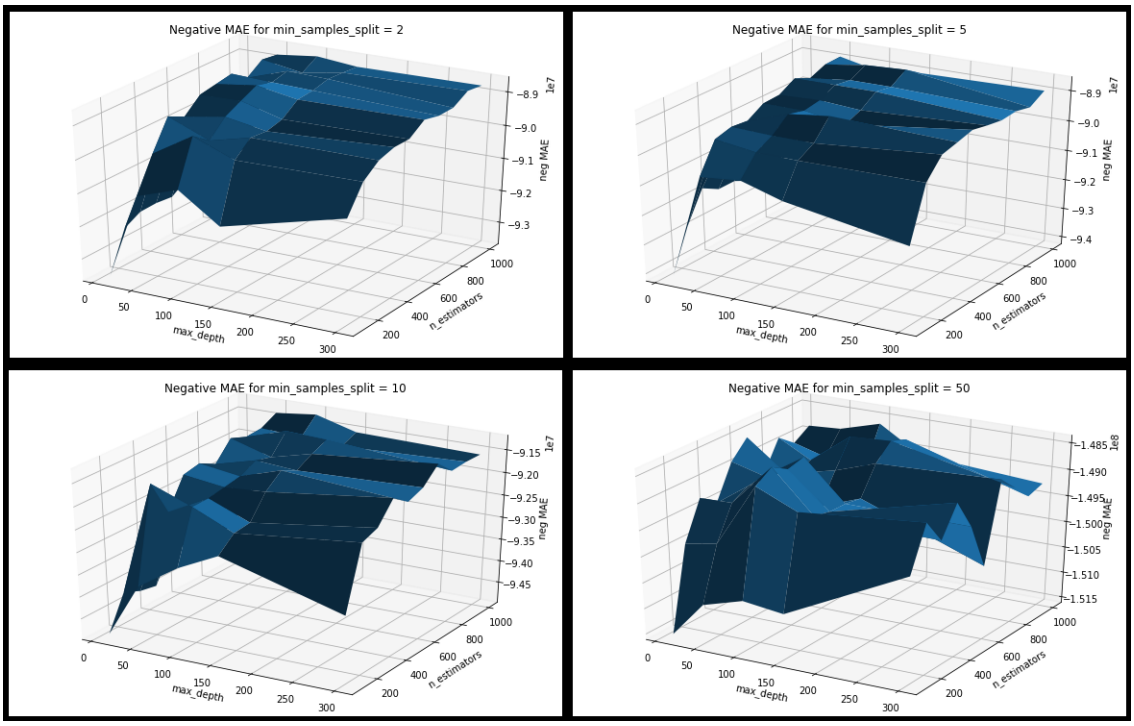
Ces modèles seront entraînés sur toute ou partie de la base d'entraînement (80% de la base totale) en fonction de la complexité et de la charge calculatoire qu'ils impliquent, et seront validés avec une 3-fold validation.

Cependant, avant de pouvoir entraîner les différents modèles sur la base finale, il a été nécessaire de déterminer les hyperparamètres optimaux de ces modèles. Afin de déterminer ces derniers, trois méthodes ont été utilisées :

— le grid-search

— le tuning itératif

— le paramétrage par avis d'expert




Évolution du score (MAE) pour le Random Forest en fonction des paramètres par grid-search

Les types de modèles retenus pour l'étude des résultats sont les suivants :

— XGBoost optimisé par grid-search sans réduction de variables

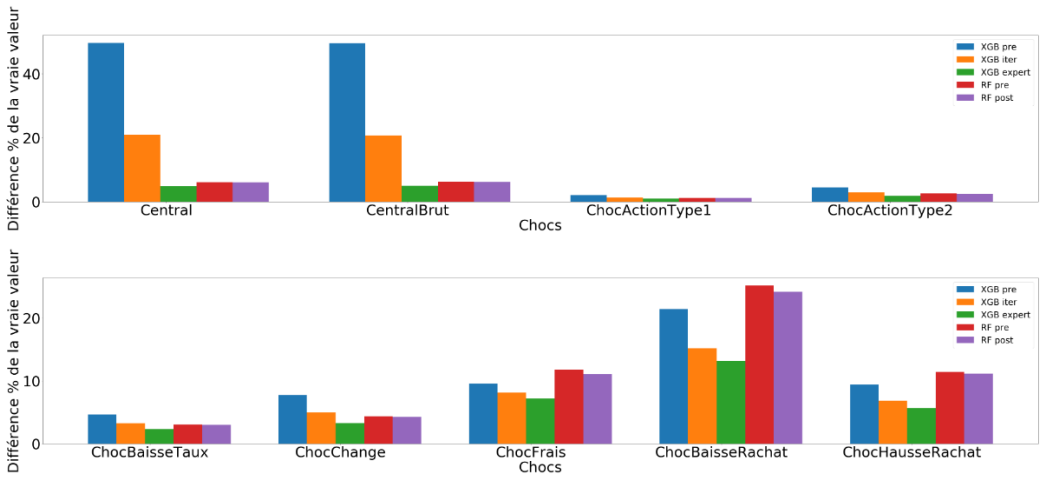
— XGBoost optimisé par avis d'expert sans réduction de variables

— XGBoost optimisé de manière itérative

— Random Forest optimisé par grid-search sans réduction de variables

— Random Forest optimisé par grid-search avec variables réduites par Recursive Feature Elimination

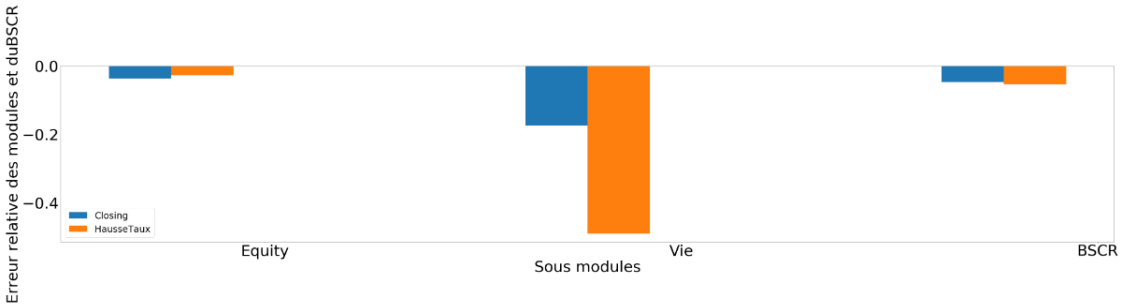
### Étude des résultats




Comparaison des erreurs relatives choc par choc pour les neufs premiers chocs

Lors de l'étude des résultats, une première analyse des erreurs relatives choc par choc permet de facilement mettre en évidence les modèles les moins performants. Deux modèles se démarquent clairement : le modèle XGBoost par avis d'expert et le modèle Random Forest optimisé par grid-search sans réduction de variables. Ces deux modèles obtiennent les meilleurs scores pour l'ensemble des chocs avec des erreurs faibles. Une étude de la distribution des résultats de ces modèles comparée aux données historiques via différents tests statistiques ne permettent pas d'identifier un modèle plus performant que les autres, mais confortent dans l'idée que les différents modèles représentent de manière précise les réponses historiques.

Finalement, étant donné que le modèle Random Forest n'était meilleur que le XGBoost que sur un seul choc et de manière minime, les derniers résultats sur les modules de SCR et le BSCR n'ont été calculés que sur le modèle XGBoost.




Erreur relative des modules de SCR et du BSCR

Le modèle XGBoost retenu permet alors d'estimer les différents SCR modulaires avec une erreur inférieure à 0.5%, et le BSCR avec une erreur inférieure à 0.2%.

## Conclusion

Un nouveau processus de calcul rapide des besoins en capital de solvabilité de l'entreprise a été mis en place dans ce mémoire. Les résultats obtenus sont très encourageants, avec des erreurs souvent inférieures à 1%. Cependant, ces travaux reposent sur de nombreuses hypothèses et informations qui peuvent être discutables, notamment sur la justesse de l'estimation de certaines variables lors d'une utilisation réelle (en particulier les variables d'IndicateurAvecPB qui ont été utilisées pour l'apprentissage). De plus, bien que les résultats théoriques soient très encourageants, du backtesting sensibilité par sensibilité reste à faire, ainsi que des modifications au niveau des variables explicatives finales afin de rendre l'outil plus accessible aux utilisateurs non experts. Malgré ces défauts, ce processus va permettre de servir de base pour les prochaines itérations et améliorations jusqu'à aboutir à un outil totalement opérationnel.

# Executive Summary

## Problematics

With Solvency 2 regulation, several techniques have been proposed in order to determine the regulatory capital for insurance companies, including the standard formula. This method allows an insurance companies to avoid to develop their own internal model, but is quite cumbersome to implement and rigid in its processes. This is the technique used by CNP Assurances to produce its SCR and therefore its solvency ratio.

However, with unstable markets, it is necessary to be able to evaluate the impact of a shock arising from sensitivities quickly and reliably in order to guide the company's management decisions. The traditional method is too slow to respond to these needs, and the techniques currently used are often not much more complex than a cross-product.

In this thesis, a new technique will be developed and tested to estimate the impact of a shock on the company's SCR using machine learning methods.

## Creation of the database

First, a database that serves both as a training and validation base for our machine learning models will be created. This database uses 40 sensitivities for the end-of-year annual closing, and contains more than 700,000 lines with more than a hundred variables. In order to create this database, a multitude of steps was necessary, from the retrieval of raw files to the transformation and cleaning of the data, in order to finally obtain a clean database containing relevant variables for future work.

| | ALLOCATION_STRATEGIQUE_Diversification | ALLOCATION_STRATEGIQUE_Taux | ... | ChocSpreadObligation | ChocSpreadTitrisation |
|---|---|---|---|---|---|
| 0 | 0.222299 | 0.733834 | ... | False | False |
| 1 | 0.222299 | 0.733834 | ... | False | False |
| 2 | 0.222299 | 0.733834 | ... | False | False |
| 3 | 0.222299 | 0.733834 | ... | False | False |
| 4 | 0.222299 | 0.733834 | ... | False | False |
| ... | ... | ... | ... | ... | ... |
| 720715 | 0.222299 | 0.733834 | ... | False | False |
| 720716 | 0.222299 | 0.733834 | ... | False | False |
| 720717 | 0.222299 | 0.733834 | ... | False | False |
| 720718 | 0.222299 | 0.733834 | ... | False | False |
| 720719 | 0.222299 | 0.733834 | ... | False | False |

720720 rows × 143 columns

Extract from the database

## Optimisation and training of the models

From this database, severals machine learning models have been tested :

— LASSO Regression

— Random Forest

— eXtreme Gradient Boost

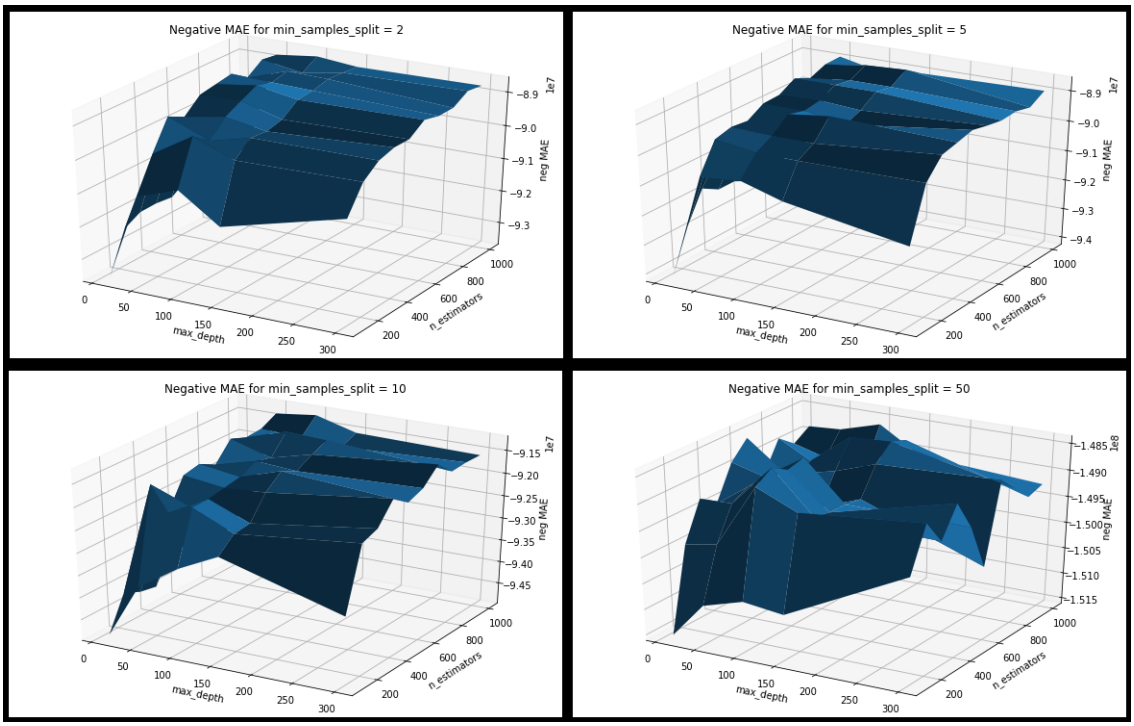
These models have been trained on all or part of the training database (representing 80% of the total database) depending on the complexity and computational load, and will be validated with a 3-fold validation.

However, before being able to trained these models on the final database, it was necessary to determine the optimal hyperparameters of these models. In order to determine these, three methods have been used :

— The grid-search

— The iterative tuning

— The expert-based parameterization




Evolution of the MAE score for the random forest depending on its parameters by grid-search

The selected models for the study of the results are the following :

— XGBoost optimised by grid-search without variables reduction.

— XGBoost with expert-based parameterization without variables reduction.

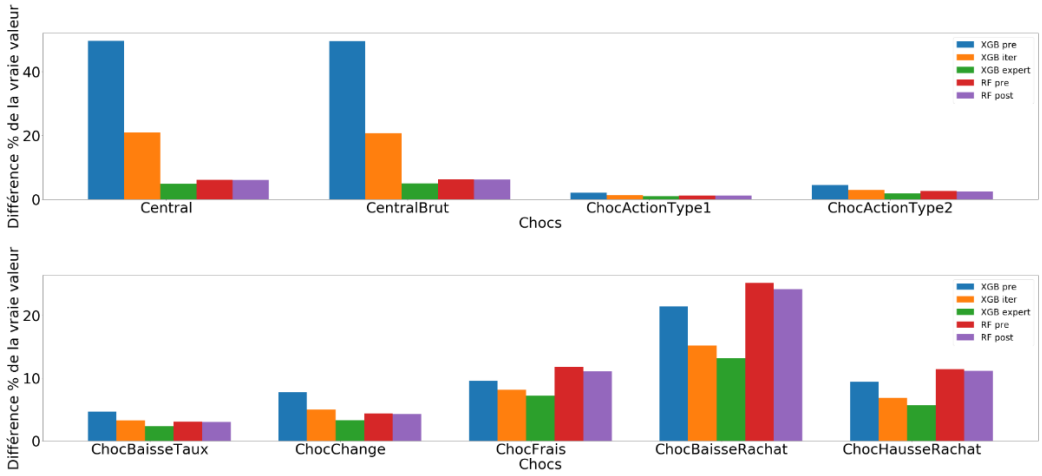
— XGBoost optimized by iterative tuning.

— Random Forest optimised by grid-search without variables reduction.

— Random Forest optimised by grid-search with Recursive Feature Elimination.

### Results' analysis

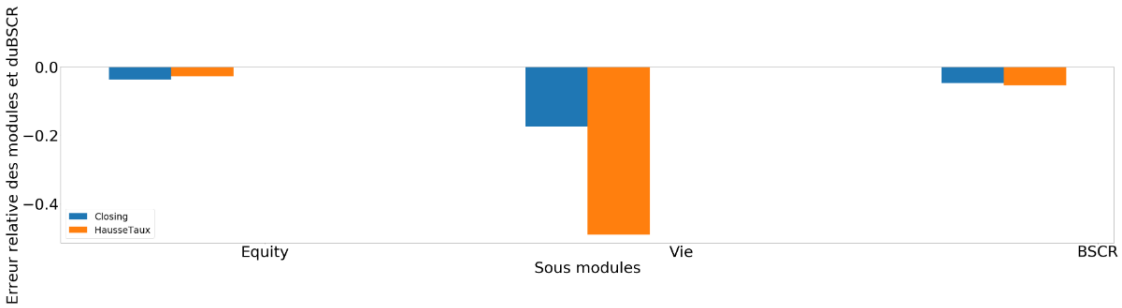
When studying the results, a first analysis of shock-by-shock relative errors makes it easy to highlight the least performing models. Two models stand out clearly : the expert-based XGBoost




Relative errors comparison shock by shock for the nine first shocks

model and the grid-search optimized Random Forest model without variable reduction. These two models obtain the best scores for all shocks with low errors. A study of the distribution of the results of these models compared to historical data through different statistical tests does not identify a more performing model than the others, but supports the idea that the different models accurately represent the historical responses.

Finally, since the Random Forest model was only better than the XGBoost model on a single shock and by only a small margin, the last results on the SCR and BSCR modules were only calculated on the XGBoost model.




Relative error on the SCR and BSCR modules

The selected XGBoost model allow to estimate the various modular SCR with a relative error lower than 0.5%, and the BSCR with an error lower than 0.2%.

## Conclusion

A new process for quickly calculating the solvency capital needs of the company has been put in place in this thesis. The results obtained are very encouraging, with errors often below 1%. However, these works are based on many assumptions and information that can be debatable, particularly on the accuracy of the estimation of certain variables in real use (especially the IndicateurAvecPB variables that were used for learning). In addition, although the theoretical results are very encouraging, sensitivity-by-sensitivity backtesting remains to be done, as well

as changes to the final explanatory variable levels in order to make the tool more accessible to non-expert users. Despite its flaws, this process will serve as a basis for the next iterations and improvements until a fully operational tool is achieved.

# Table des matières

**Introduction** **14**

**1 La réglementation Solvabilité II** **16**

1.1 Évolution de la vision des engagements de l'assureur . . . . . 16

1.2 Les différents piliers . . . . . 18

1.2.1 Pilier 1 : Pilier Quantitatif . . . . . 18

1.2.2 Pilier 2 : Pilier Qualitatif . . . . . 20

1.2.3 Pilier 3 : Reporting . . . . . 21

1.3 Les méthodes de calcul du capital économique sous Solvabilité 2 . . . . . 21

1.3.1 Formule standard . . . . . 22

1.3.2 Modèle interne et modèle hybride . . . . . 25

**2 Base de données** **26**

2.1 Production de SCR sous formule standard chez CNP Assurances . . . . . 26

2.1.1 Description du portefeuille d'actifs . . . . . 27

2.1.2 Description du portefeuille d'assurés . . . . . 28

2.2 Création de la base de données . . . . . 30

2.2.1 Traitement et uniformisation des variables . . . . . 30

2.2.2 Récupération des données . . . . . 31

2.2.3 Traitement des données . . . . . 31

2.2.4 Transformation des formats . . . . . 34

2.2.5 Suppression des variables non pertinentes . . . . . 36

2.2.6 Base de données finale et analyse des données . . . . . 37

**3 Application des modèles de Machine Learning et réduction des dimensions** **43**

3.1 Introduction au Machine Learning ou apprentissage automatique . . . . . 43

3.2 Présentation des différents modèles . . . . . 44

3.2.1 Arbre de classification et de régression . . . . . 45

3.2.2 Random Forest . . . . . 47

3.2.3 eXtreme Gradient Boosting . . . . . 50

3.3 Application et tuning des hyperparamètres . . . . . 51

3.3.1 Le Grid Search . . . 52
3.3.2 Tuning itératif et algorithmes gloutons . . . 53
3.3.3 La Validation Croisée . . . 54
3.3.4 Random Forest . . . 56
3.3.5 eXtreme Gradient Boosting . . . 62
3.4 Réduction de la dimension et sélection de variables . . . 67
3.4.1 Régression LASSO . . . 68
3.4.2 Recursive Feature Elimination (RFE) . . . 69

**4 Résultats, interprétation et critiques** **74**
4.1 Études des résultats . . . 75
4.2 Étude de la distribution de la variable cible et des prédictions . . . 79
4.2.1 Comparaison avec des distributions existantes . . . 79
4.2.2 Kolmogorov Smirnov Test . . . 83
4.2.3 *t*-test de Student . . . 85
4.2.4 Critère de Cramér-Von Mises . . . 86
4.3 Agrégation des SCR et erreur finale . . . 88
4.4 Limites et critiques . . . 91
4.4.1 Limitations physiques . . . 91
4.4.2 Limitations algorithmiques et liées aux modèles . . . 92
4.4.3 Limitations liées aux processus et données utilisées . . . 93
4.5 Pistes d'explorations potentielles . . . 94
4.5.1 Autres modèles de machine learning et deep learning . . . 94
4.5.2 Utilisation de modèles de machine learning pour la classification . . . 95

**5 Conclusion générale** **96**

# Introduction

Les activités des compagnies d'assurance sont régulées de manière très stricte par différentes directives et réglementations.

La directive Solvabilité 2 (surnom de la Directive 2009/138/CE du Parlement européen et du Conseil du 25 novembre 2009), entrée en application le 1er Janvier 2016, fait suite à Solvabilité 1, cette dernière étant apparue dans les années 70. Cette nouvelle directive donne une vision du risque évoluée, plus axée sur une vision valeur de marché et requiert de nouveaux seuils de capitaux minimum pour rester solvables. Ces seuils, le MCR et surtout le SCR, peuvent être calculés de manière diverse. La formule standard est la méthode la plus simple mais ne permet pas une bonne prise en compte du profil de risque de l'assureur, tandis que le modèle interne, le plus complexe, est très lourd et coûteux à mettre en place. Peu importe la méthode choisie, le calcul du SCR requiert de très importantes ressources et génère des temps de calcul pouvant être longs. Bien que cela ne soit pas très important dans le cadre d'une production annuelle ou trimestrielle du SCR, cela devient très limitant dans les cas où nous voulons effectuer des stress-tests ou mesurer l'impact d'une crise afin de pouvoir réagir rapidement.

L'objectif de ces seuils de solvabilité est d'apporter un contrôle plus minutieux des activités de ces entreprises et de faire en sorte que les risques pris soient correctement évalués et que les compagnies d'assurance disposent de suffisamment de ressources pour faire face à diverses crises.

Dans le cadre du modèle interne, et plus précisément pour les entreprises utilisant la méthode des Simulations dans les Simulations (SdS), il existe de nombreuses méthodes afin d'accélérer ces calculs (LSMC, krigeage stochastique, méthode Loisel, ...), étant donné qu'elle est beaucoup plus flexible que la formule standard. La formule standard étant beaucoup moins polyvalente dans son implémentation, les méthodes de Machine Learning ne font pas partie des méthodes utilisées pour la production, mais il est possible d'obtenir des estimations du SCR via ces outils.

La formule standard est généralement utilisée par des entreprises n'ayant pas une activité suffisamment importante ou les moyens pour justifier le développement d'un modèle interne. CNP Assurances fait ici exception, étant le leader de l'assurance emprunteur et le second assureur vie en France avec une activité très importante, nous sommes amenés à penser que l'utilisation du modèle interne serait privilégiée par l'entreprise afin de mieux cerner ses risques et réduire son besoin en capital. Cependant, la formule standard est suffisamment cohérente avec les risques pris par CNP Assurances pour que le développement d'un modèle interne ne faisait pas sens.

Afin de déterminer son niveau de SCR, CNP Assurances utilise NEMO (Nouvel Environnement de Modélisation) qui est l'outil interne développé pour appliquer la formule standard à ses portefeuilles. Cet outil à notamment l'avantage de produire des données à des niveaux de granularité variables dans le processus de calcul du SCR et permet donc d'obtenir des données et informations de très grandes qualitées et en grand nombre.

Dans des contextes où les marchés ne sont pas stables et peuvent se retourner à tout moment,

il est nécessaire de disposer de techniques ou de processus permettant d'estimer l'impact sur le ratio de l'entreprise de ces changements imminents le plus rapidement possible afin que les dirigeants puissent prendre des choix rapidement. Les méthodes classiques de calcul du SCR ne répondent pas à ce critère de vitesse et les assureurs sont donc amenés à développer des outils plus efficaces pour de tels scénarios, quitte à perdre de la précision sur le ratio final.

Parmi ces outils, les méthodes de machine learning sont très appréciées, étant à la fois très rapides et performantes. L'inconvénient est le plus souvent le manque de transparence dans le résultat et donc la perte d'interprétabilité qui serait pénalisant dans le contexte d'une publication de ratio officielle, cependant pour un usage interne cela pose alors moins de problèmes.

Dans le cas de CNP Assurances, il est donc question de développer un outil basé sur le machine learning afin de répondre à cette problématique. Une première approche a été effectuée par Tarek AOUDI dans son mémoire chez CNP Assurances mais qui était plus focalisée sur la minimisation du SCR. Dans le cadre de ce mémoire, une base de données sera complètement reconstruite afin de maîtriser complètement les choix effectués et surtout de la documenter pour qu'elle soit réutilisable et améliorable par de futurs travaux.

Les données officielles de CNP Assurances seront donc récupérées afin de former un socle idéal pour construire, optimiser et tester des modèles de machine learning par-dessus qui donneront la possibilité d'effectuer des estimations de SCR en modifiant différentes hypothèses.

# 1 La réglementation Solvabilité II

La directive Solvabilité II a pris effet le 1er Janvier 2016. Cette directive vise les compagnies d'assurance de l'Union Européenne en y apportant son lot d'évolutions et de nouvelles réglementations par rapport à son prédécesseur Solvabilité I.

Les premières versions de Solvabilité 1 ont été adoptées dans les années 70 et la directive a été complétée notamment avec un volet assurance vie en 2002. Vieillissante, elle a rapidement montré ses limites, notamment avec les crises financières majeures de 2002-2003 et de 2007-2008. Bien qu'ayant l'avantage d'avoir des directives plus simples à comprendre et à appliquer, elle ne permettait pas de prendre en compte le profil de risque de l'assureur de manière fidèle et permettait à certains de prendre des risques considérables non pénalisés par la réglementation en vigueur.

Les travaux qui mèneront à Solvabilité II sont alors rapidement entamés pour être votés en Avril 2009. Plus qu'une évolution, cette nouvelle directive va faire table rase de la vision des engagements des assureurs en Europe, et va venir définir un cadre complet, à la fois quantitatif et qualitatif, afin de s'assurer de la solvabilité des compagnies concernées.

Ce changement ne s'est évidemment pas fait du jour au lendemain, avec de nombreuses études ayant été effectuées avant, pendant et après la mise en place de cette directive, et en laissant un temps d'adaptation conséquent aux compagnies d'assurance (7 ans entre l'adoption de la loi et la prise d'effet).

## 1.1 Évolution de la vision des engagements de l'assureur

Un des grands changements apportés par la directive Solvabilité II est la modification de la manière de considérer les fonds propres de l'assureur, en les évaluant en "valeur de marché" et en demandant un niveau de fonds propres permettant d'éviter la ruine avec une probabilité de 99.5% sur un an.

Afin de calculer ce niveau de fonds propres nécessaire, les risques sont divisés en modules, eux-mêmes divisés en sous modules et le niveau de capital requis est ensuite calculé, avec différentes méthodes possibles que nous détaillerons par la suite.

Sous Solvabilité I, les compagnies d'assurance n'évaluaient leur bilan que de manière comptable, ne prenant alors pas ou peu en compte la réalité économique dans laquelle elles se situaient. La directive demandait alors un niveau de capital nécessaire pour être solvable bien différent de celui obtenu par Solvabilité II.

Dans le cadre de Solvabilité I, le besoin en capital était déterminé par l'Exigence de Marge de Solvabilité (EMS) qui dépend de différents paramètres tels que les primes touchées, le montant des sinistres ou des prestations et des provisions mathématiques. Dans le cas d'une assurance-vie, on peut résumer cela à un pourcentage des provisions mathématiques des produits en euros et des produits en unités de compte (UC), ainsi qu'à une somme dépendant des capitaux sous risque et de la durée des engagements.

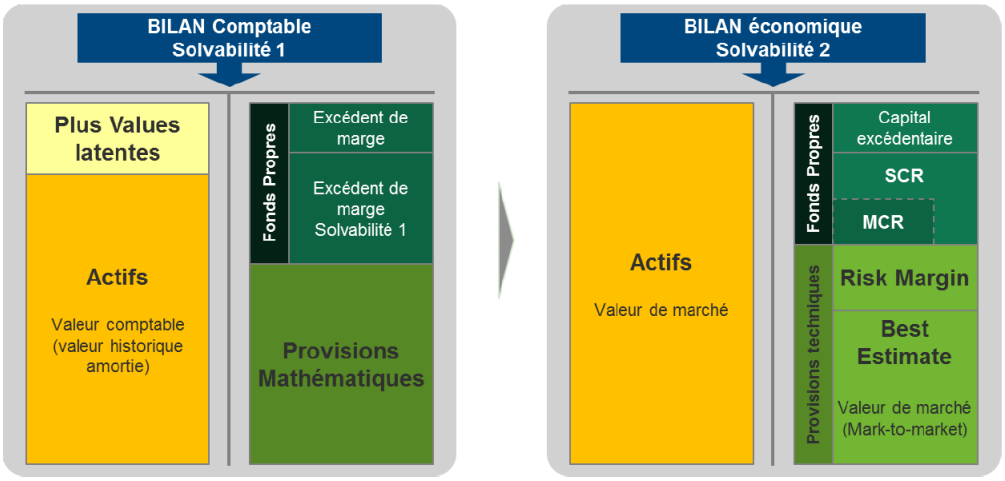



FIGURE 1 – Comparaison du bilan comptable sous Solvabilité I au bilan économique sous Solvabilité II

Source: InsuranceSpeaker

$$EMS = 4\% \times PM_{euro} + 1\% \times PM_{UC} + \alpha_t CSS \tag{1}$$

Avec :

— $PM_{euro}$ (resp $PM_{UC}$) : les provisions mathématiques provenant de produits en Euro (resp UC)

— CSS : les capitaux sous risques

Il est alors assez clair qu'il y a un manque de prise en compte des profils de risques des assureurs. En effet, l'EMS ne prend pas du tout en compte l'origine des provisions techniques et des capitaux sous risque.

En effet, supposons deux assureurs A et B ne commercialisant que des contrats sur des supports en euros. Le premier assureur A investit l'intégralité de ses primes dans des actions tandis que l'assureur B investit en totalité dans des obligations souveraines. Sous la loupe de Solvabilité 1, les EMS pour les deux assureurs seront égaux bien qu'il soit assez clair que les risques supportés sont bien différents. Les placements sur actions seront beaucoup plus risqués que ceux sur les obligations souveraines mais l'investissement ne sera pas plus pénalisé. Il y a alors un risque que les assureurs profitent de cette asymétrie pour investir sur des supports plus risqués en vue de rendements plus élevés au détriment du risque supplémentaire encouru.

Dans le cadre de Solvabilité II, le bilan ainsi que les provisions techniques sont calculés en valeur de marché. Cela permet alors de prendre en compte les différentes dynamiques de risques sous-jacentes à chaque produit, et donc de pénaliser de manière différenciée les placements et donc les profils de risque des assureurs. Sous Solvabilité II, le SCR pouvant être vu comme un niveau de perte catastrophique de probabilité 0.05%, l'assureur A aurait alors un SCR bien plus élevé que l'assureur B. Son investissement étant plus risqué, la perte pour la même probabilité de 0.05% sera supérieure et ce mécanisme incite donc à gérer minutieusement les risques auxquels les assureurs sont exposés.

## 1.2 Les différents piliers

La directive Solvabilité II a été construite autour de 3 piliers directeurs. Chacun de ces piliers joue un rôle clef dans le processus de contrôle continu de la solvabilité des assureurs.
Le premier pilier va regrouper toutes les exigences quantitatives et règles qui vont être demandées aux assureurs : hypothèses sur les actifs et les passifs, exigences en capital et modes de calcul de ces dernières.
Le second pilier va concerner les exigences qualitatives, que ce soit sur les règles de gouvernance, sur leurs politiques de gestion des risques ou sur leurs propres évaluations de leurs risques et solvabilité (Own Risk and Solvency Assessment, i.e, ORSA).
Le troisième et dernier pilier donne l'ensemble des exigences vis à vis de la communication de l'information au public et aux autorités de contrôle (e.g, l'ACPR en France). Cela passe notamment par la publication de reportings standardisés afin d'obtenir des documents comparables et harmonisés au niveau européen.

### 1.2.1 Pilier 1 : Pilier Quantitatif

Sous le pilier 1, nous retrouvons l'ensemble des exigences quantitatives :

- Minimum de Capital Requis (Minimum Capital Requirement) : MCR
- Capital de Solvabilité Requis (Solvency Capital Requirement) : SCR
- Hypothèses sur les courbes des taux sans risques
- Ajustement pour la volatilité (volatility adjustment - VA)
- Ajustement égalisateur (matching adjustment - MA)
- et autres ...

Le MCR et le SCR sont deux valeurs clefs pour les compagnies d'assurance. Ces niveaux de capitaux sont régulièrement contrôlés et un niveau de capital insuffisant peut mener à la cessation d'activité de l'entreprise.

Le MCR (article R352 du Code des Assurances) correspond au niveau minimum de fonds propres de base éligibles qu'une société d'assurance doit détenir afin de pouvoir répondre à ses engagements envers les assurés, souscripteurs et bénéficiaires des contrats. En dessous de ce niveau, l'autorité de contrôle intervient et si leur niveau de fonds propres ne remonte pas au-dessus de ce seuil minimum, l'agrément d'assurance peut leur être retiré.

Le MCR est donc le niveau minimal de fond en dessous duquel les assurés et autres parties preneurs de risques seraient exposés à un niveau de risque inacceptable. Ce MCR ne peut pas être inférieur à 2.2 millions d'euros pour les entreprises d'assurance non-vie et à 3.2 millions d'euros pour les assureurs vie.

La formule pour déterminer le MCR est relativement simple : elle correspond à la VaR (Value At Risk) de la valeur en risque des fonds propres pour un niveau de confiance de 85% à l'horizon d'un an. De plus, elle ne doit pas être inférieure à 25%, ni être supérieure à 45% du SCR de l'entreprise.

Le SCR représente le capital nécessaire pour éviter la ruine à un horizon de 1 an avec une probabilité de 99.5%. Ce dernier doit être calculé à minima une fois par an et communiqué aux autorités de contrôle.

Afin de définir plus rigoureusement le SCR, nous avons besoin de définir les termes capital économique et ruine économique. Sous la réglementation Solvabilité 2, le capital économique est le montant minimum des fonds propres qui permet d'éviter la ruine économique sur un horizon de 200 ans. On dit que la compagnie d'assurance est en ruine économique si ses fonds propres deviennent négatifs en vue économique (de marché).

Si nous notons $CD_i$ le capital disponible en $t = i$, alors nous avons que le capital requis $x$ pour éviter une telle ruine est tel que :

$$\mathbb{P}(CD_1 > 0 | CD_0 = x) \geq 99,5\%$$

La probabilité conditionnelle ci-dessus est évidemment croissante et continue avec $x$. Il existe donc un plus petit $x$ qui satisfait notre condition sur le niveau de confiance.

Ce minimum correspond alors à notre SCR :

$$SCR = \underset{x \in \mathbf{R}}{arg\,min} \mathbb{P}(CD_1 > 0 | CD_0 = x) \geq 99,5\%$$

Ou encore en considérant $L = CD_0 - \frac{CD_1}{1+r}$, qui représente alors la perte de capital disponible entre l'année 1 et l'année 0 avec $r$ le taux d'actualisation, on a :

$$SCR = \underset{x \in \mathbf{R}}{arg\,min} \mathbb{P}(L > x) \leq 0,5\%$$

et donc on a :

$$SCR = VaR_{99.5\%}(L)$$

Il existe de plus un lien entre la norme MCEV et Solvabilité 2. Dans la première, les fonds propres sont composés de l'Actif Net Reévalué (ANR) et la Value In Force (VIF). L'ANR est composé des actifs non dépendants des engagements techniques des assurés, évalués à leurs valeurs de marché, tandis que la VIF correspond à la valeur actuelle du portefeuille (et donc de ses gains futurs) sans prendre en compte les futures entrées et sorties du portefeuille. Sous Solvabilité 2, les fonds propres sont constitués du SCR/MCR ainsi que du capital de surplus excédentaire.

Nous avons alors les relations suivantes :

$$ANR + VIF = VM - (BE + RM)$$

et donc :

$$SCR = \Delta MCEV = \Delta ANR + \Delta VIF$$

Plus précisément, il est calibré de sorte à *"garantir que tous les risques quantifiables auxquels l'entreprise d'assurance ou de réassurance est exposée soient pris en considération"*. Ce niveau de capital requis constitue un point majeur pour les compagnies d'assurance, aussi bien à cause de sa complexité à produire que du reflet de la santé de l'entreprise qu'il représente. Elle correspond donc au montant de fonds propres que l'assureur doit immobiliser afin d'absorber les chocs potentiellement provoqués par des événements et situations exceptionnelles. Cela garantit alors la capacité de l'assureur d'honorer ses engagements, en particulier ceux qui les lient à ses assurés.

Pour calculer le SCR d'une entreprise, plusieurs méthodes existent mais les plus courantes sont le modèle interne et la formule standard.

Ce SCR sert au final à calculer le taux de couverture de l'entreprise. Ce taux de couverture correspond au rapport entre les fonds propres éligibles de l'entreprise et le capital requis :

$$Ratio = \frac{FP\,Eligibles}{Capital\,Requis}$$

Ce ratio permet de mesurer la solvabilité de l'entreprise de manière assez simple. Plus ce ratio est haut, plus l'entreprise dispose de capital en excédent permettant de couvrir le MCR et représente un signe de bonne santé et solvabilité de l'entreprise.

Il est donc attendu que les assureurs aient un ratio supérieur à 100%. Dans le cas contraire, le régulateur serait dans l'obligation d'intervenir car l'entreprise ne serait pas en capacité de répondre à ses engagements d'assureur.

### 1.2.2 Pilier 2 : Pilier Qualitatif

Le pilier 2 de la directive concerne l'exigence de gouvernance et de gestion des risques ainsi que du processus ORSA.

Cela va donc notamment passer par la mise en place d'un système de gestion des risques de manière formelle (règles ERM, Enterprise Risk Management), qui devra pouvoir être contrôlé par le régulateur, et dans les cas où il serait jugé insuffisant, un capital supplémentaire public appelé "capital add-on" pourra être demandé.

Le processus ORSA (Own Risk and Solvency Assessment) de l'article 45 de la directive Solvabilité II nécessite aux entreprises de définir un ensemble de procédures afin d'inclure le risque de l'entreprise dans la prise de décisions et l'analyse stratégique de l'entreprise. Elle vise à évaluer de manière continue le risque pris par une entreprise afin d'évaluer la solvabilité de cette dernière à tout instant, en prenant en compte la spécificité de son profil de risque.

L'entreprise va donc être amenée à définir son profil de risque, c'est à dire à quels risques est-elle exposée et à quel point est-elle exposée à ces risques. Après avoir déterminé son profil de risque, l'entreprise devra définir son appétit au risque ainsi que sa tolérance au risque. C'est donc communiquer à quel point est-elle prête à prendre des risques par rapport aux gains potentiels.

De manière naturelle, le processus d'ORSA va de pair avec le 3eme pilier de la directive Solva-

bilité II portant sur la production de reporting, mais elle devra aussi faire l'objet de son propre reporting destiné à l'administration et au superviseur.

### 1.2.3 Pilier 3 : Reporting

Le troisième et dernier pilier constituant Solvabilité 2 dicte les consignes en terme de reportings et de communication avec le marché (l'information mise à disposition du public). Il existe alors un certain nombre de reportings à effectuer et soumettre selon un planning réglementé. Une partie de ces rapports sont à destination du superviseur seul, mais d'autres doivent être publiés publiquement.

On distingue ces reportings en deux catégories : les états quantitatifs et les reportings narratifs.

Les états quantitatifs :

— Les Quantitative Reporting Templates (QRT, reportings quantitatifs) sont des rapports devant être produits par l'ensemble des assureurs des pays européens soumis à Solvabilité 2. D'échéance trimestrielle et annuelle, ces rapports demandent des analyses sur de nombreux secteurs différents (provisions techniques, SCR, MCR, fonds propres, ...) en vision solo et en vision groupe. Il en existe des publics et à destination du superviseur seul.

— Les Etats Nationaux Spécifiques (ENS) ne concernent que le marché français et sont définis par l'ACPR. Ils reprennent principalement des éléments qui étaient communiqués par les assureurs sous Solvabilité I. Il s'agit donc d'une continuité des reportings de Solvabilité I dans Solvabilité II. Ces reportings sont à destination exclusive du superviseur et ne sont donc pas communiqués au public.

— Et d'autres tels que les FSR et les états BCE.

Les reportings narratifs :

— Les Solvency and Financial Condition Report (SFCR, rapport sur la solvabilité et la situation financière) sont des rapports exigés par le pilier 3 de la directive Solvabilité II devant être publiés tous les ans. Ce sont des rapports à destination du superviseur et du public. Ces documents présentent l'activité et les résultats de l'entreprise, en décrivant ses différentes performances techniques et financières. Elles donnent aussi des indications quant à la direction prise en terme de gestion des risques et du profil de risque de l'entreprise.

— Les Regular Supervisory Report (RSR, rapport régulier au contrôleur) est un rapport devant être publié au minimum tous les 3 ans qui n'est à destination que du superviseur.

## 1.3 Les méthodes de calcul du capital économique sous Solvabilité 2

Nous avons précédemment introduit la réglementation Solvabilité 2 et notamment son premier pilier, le pilier quantitatif. Ce pilier quantitatif regroupe de nombreux indicateurs mais celui qui va nous intéresser par la suite est le SCR. Ce dernier peut être calculé de diverses manières et nous allons en présenter quelques-unes. Par la suite, nous nous focaliserons sur la formule standard, étant la méthode retenue par CNP Assurances pour calculer son SCR.

### 1.3.1 Formule standard

La formule standard est, comme son nom l'indique, standardisée pour être applicable à tous les assureurs. Elle vise à refléter le profil de risque de la plupart des assureurs et réassureurs. De ce fait, il peut arriver qu'elle ne traduise pas correctement le profil de risque particulier d'une entreprise. Il existe donc des recours afin de pouvoir adapter certains paramètres en donnant la possibilité aux entreprises d'utiliser leurs propres données et calibrage pour les modules de risque de souscription.

Pour calculer le $SCR_{global}$, la formule standard se base sur des modules de SCR, qui eux-mêmes se décomposent en sous modules, et de chocs. Cela permet de classer chaque produit et flux dans une de ces cases, et appliquer les chocs associés à chaque sous module avant de les agréger et d'obtenir le $SCR_{global}$ recherché.

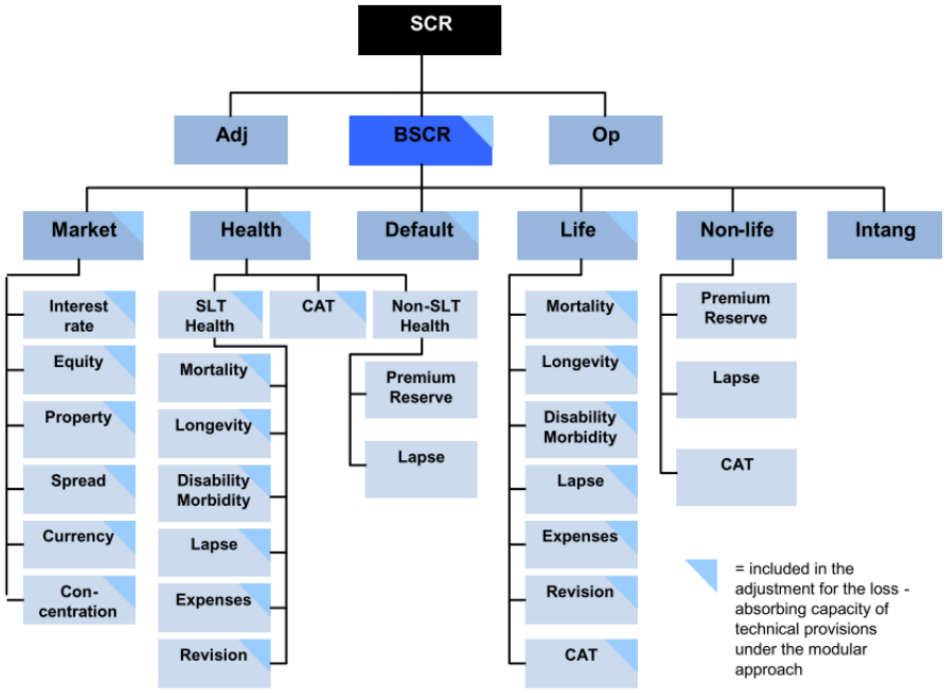



FIGURE 2 – Structure modulaire pour le calcul du SCR global sous formule standard

Source: EIOPA

Le $SCR_{global}$ se décompose alors de la manière suivante :

$$SCR_{global} = BSCR + SCR_{op} + Adj \tag{2}$$

Avec :

— BSCR (Basic Solvency Capital Requirement) : Somme de l'agrégation des SCR modulaires et du SCR des actifs intangibles. L'agrégation est faite à l'aide de matrices de corrélations.

— Adj : Ajustement calculé via le BSCR et nBSCR (BSCR net). Le nBSCR va inclure la capacité d'absorption des pertes par les provisions techniques et les impôts différés.

— $SCR_{op}$ : Correspond au module de risque opérationnel

Le BSCR va agréger les différents modules de risques par la formule suivante :

$$BSCR = \sqrt{\sum_{i,j} \rho_{i,j} \times SCR_i \times SCR_j + SCR_{intangibles}} \tag{3}$$

avec :

— $(i, j)$ parcourant l'ensemble des couples de modules

— $\rho_{i,j}$ le coefficient de corrélation entre les modules $i$ et $j$

— $SCR_i$ le SCR du module $i$

— $BSCR$ l'agrégation des différents SCR modulaire

La matrice de corrélations appliquée pour le calcul du BSCR est une matrice réglementaire. Les valeurs de cette matrice peuvent être discutées, ne représentant pas forcément la réalité économique sous-jacente.

| $CorrSCR$ | $SCR_{mkt}$ | $SCR_{def}$ | $SCR_{life}$ | $SCR_{health}$ | $SCR_{non-life}$ |
|---|---|---|---|---|---|
| $SCR_{mkt}$ | 1 | | | | |
| $SCR_{def}$ | 0.25 | 1 | | | |
| $SCR_{life}$ | 0.25 | 0.25 | 1 | | |
| $SCR_{health}$ | 0.25 | 0.25 | 0.25 | 1 | |
| $SCR_{non-life}$ | 0.25 | 0.5 | 0 | 0.25 | 1 |

FIGURE 3 – Matrice des corrélations entre les SCR modulaires

Le SCR doit alors couvrir au minimum les risques suivants :

— le risque de souscription en non-vie

— le risque de souscription en vie

— le risque de souscription en santé

— le risque de marché

— le risque de contrepartie

— le risque opérationnel

Comme nous pouvons le voir dans la figure 2, chaque module de risque est décomposé en sous modules. Le principe est ici le même, le capital sous-modulaire est calculé de manière individuelle et est ensuite agrégé en prenant en compte les potentiels effets de diversification.

Nous obtenons alors la formule suivante pour l'agrégation sous modulaire :

$$SCR_m = \sqrt{\sum_{i,j \in R_m} \rho_{i,j}^{R_m} \times C_i \times C_j} \tag{4}$$

avec :

— $R_m$ : l'ensemble des risques sous modulaires du module $m$

— $\rho_{i,j}^{R_m}$ : les coefficients de corrélations entre les risques $i$ et $j$ du module $R_m$

— $C_i$ : le capital du sous-module $i$.

— $SCR_m$ : le capital requis pour le module $m$

Chaque module a de même sa propre matrice de corrélations afin d'agréger les résultats de ses sous modules.

Le calcul du SCR sous formule standard se base sur une approche par scénarios. Nous disposons d'un scénario central (typiquement les données actuelles projetées sur une certaine période) et nous allons considérer un certain nombre de scénarios stochastiques qui vont nous permettre d'avoir une vision probabiliste de l'état du marché dans le futur. Chacun de ces scénarios vont nous permettre d'évaluer l'impact d'un choc sur les fonds propres de l'entreprise, et à l'aide d'un assez grand nombre de scénario, il est possible d'en estimer une répartition simulée des fonds propres et donc du SCR lié à ce choc.

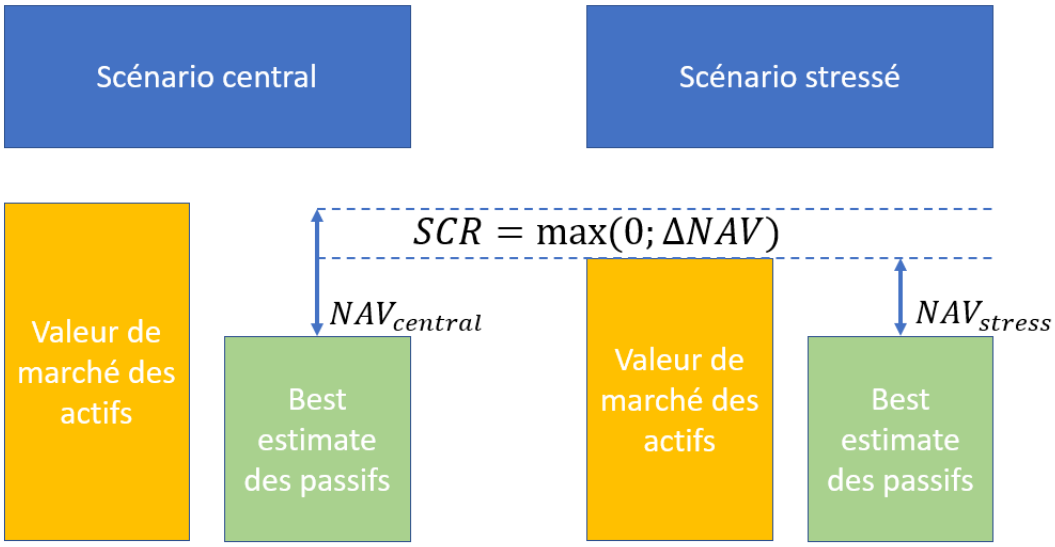



FIGURE 4 – Méthode $\Delta NAV$

On a donc que pour un sous module donné $m$ :

$$\begin{aligned} SCR_m &= \Delta NAV \\ &= NAV_{central} - NAV_{choc} \\ &= (VM_{central} - BE_{central}) - (VM_{choc} - BE_{choc}) \\ &= \Delta VM - \Delta BE \end{aligned}$$

Bien que cette méthode soit le standard adopté par la réglementation, de nombreuses critiques ont été faites à son propos. Une des plus citée est celle en rapport avec la linéarité de la corrélation entre les facteurs de risques, que ce soit au niveau modulaire ou sous modulaire. Dans le cas des modèles internes ou hybrides, des modèles de corrélation plus complexes peuvent être mis en place à l'aide de copules.

### 1.3.2 Modèle interne et modèle hybride

L'entreprise d'assurance ou de réassurance peut effectuer une demande auprès des autorités de contrôle afin d'utiliser un modèle interne pour calculer leur capital économique. Ce passage vers un modèle interne peut aussi être forcé par l'autorité dans le cas où le profil de risque de l'entreprise s'écarte significativement des hypothèses sous-jacentes de la formule standard.

Pour ce type de calcul, l'ensemble des interactions entre les actifs, passifs, portefeuilles d'assurances ainsi que leur modélisation doit être pris en compte. Bien plus complexe à mettre en oeuvre, il permet néanmoins d'appliquer des méthodes diverses afin d'accélérer la production du SCR de l'assureur, ainsi que de modifier certaines hypothèses qui seraient fixées sous la formule standard.

La majorité des modèles internes utilisent des modèles et techniques de simulations dits stochastiques. Afin de représenter au mieux possible les risques encourus, certains indicateurs vont être tirés aléatoirement en suivant une certaine distribution de probabilité afin d'obtenir une vision probabiliste des risques souscrits.

Le modèle hybride quant à lui reprend des calculs de la formule standard mais permet l'application d'un modèle interne sur certains points spécifiques nécessitant une approbation du régulateur. Cela permet de ne pas avoir à implémenter un modèle interne sur l'ensemble de ses secteurs d'activité et risques mais uniquement sur ceux dont les hypothèses divergent le plus de la réalité de l'entreprise et qui leur coûtent donc le plus cher.

# 2 Base de données

Avant de commencer la création de la base de données, une présentation du processus de production du SCR chez CNP Assurances (simplifiée) sera effectuée pour comprendre les différents mécanismes en jeu.

## 2.1 Production de SCR sous formule standard chez CNP Assurances

NEMO (Nouvel Environnement de MOdélisation) est l'outil utilisé par CNP Assurances afin d'effectuer ses différents calculs et simulations. Cet outil permet de centraliser les tâches et actions dans un même logiciel et offre la possibilité de créer des sensibilités en connectant des entrées spécifiques à des algorithmes.

Une vision simplifiée pour la production d'un SCR peut être vue de la manière suivante :

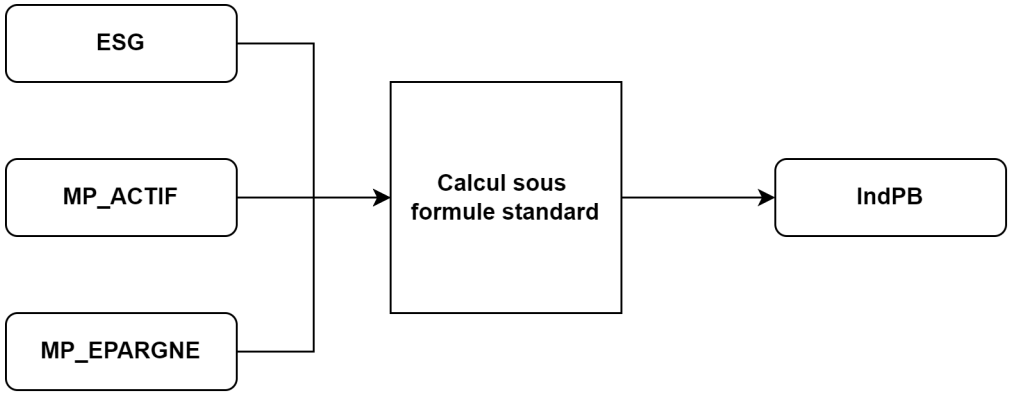



FIGURE 5 – Diagramme simplifié de la production d'un SCR sous NEMO

Les données que nous utiliserons seront alors les entrées (à gauche) et sorties (à droite) de cet environnement :

— **ESG** (Economic Scenario Generator) : Données du marché contenant 1000 scénarios différents et existant en version Up, Down et Central. La version centrale ne comporte qu'un seul et unique scénario en déterministe, tandis que les deux autres correspondent respectivement aux scénarios à la hausse et à la baisse

— **MP_EPARGNE** : Données du model point de passif, contenant entre autre des informations sur le portefeuille de contrat, l'âge et l'ancienneté des assurés ainsi que les provisions mathématiques attachées.

— **MP_ACTIF** : Données du model point d'actif, contenant les différentes lignes du portefeuille d'actifs, le type d'actif, leurs valeurs boursières et autres.

— **IndPB** : Données de sortie des calculs de NEMO contenant entre autre l'Equity (correspondant à la VIF du chapitre 1).

Les trois entrées sont alors transférées au bloc de calcul qui va dérouler toutes les étapes du calcul ALM jusqu'à obtenir nos différentes sorties IndicateursAvecPB.

Chaque sortie *IndPB* de ce bloc de calcul comporte 18 sous sorties de type $IndPB_i$ avec $i$ allant de 1 à 18. Ces 18 sous sorties correspondent aux différents chocs appliqués et que nous allons exploiter pour construire notre base de données.

TABLE 1 – Liste des 18 chocs appliqués

| Central | ChocHausseTaux | ChocChange |
|---|---|---|
| CentralBrut | ChocImmobilier | ChocFrais |
| ChocActionType1 | ChocLongevite | ChocHausseRachat |
| ChocActionType2 | ChocMortalite | ChocSpreadCDS |
| ChocBaisseRachat | ChocMortaliteCat | ChocSpreadObligation |
| ChocBaisseTaux | ChocRachatMassif | ChocSpreadTitrisation |

Les données utilisées par CNP Assurances dans la formule standard leurs permettent de produire différents résultats dont l'*Equity* qui nous permettra d'approcher et de déduire le SCR final.

### 2.1.1 Description du portefeuille d'actifs

Avant de nous attaquer à la création de la base de données qui servira à entraîner nos modèles, nous allons décrire rapidement la composition des portefeuilles sur lesquels nous travaillerons. Ces descriptions seront faites sur les portefeuilles centraux du Closing du T4 2021 de produits d'épargne en euros. Ces portefeuilles varient bien sûr avec le temps mais restent relativement stables dans le temps.

Le portefeuille d'actifs a une valeur initiale de 384 896 376 213€ et est composé de plusieurs types d'actifs :

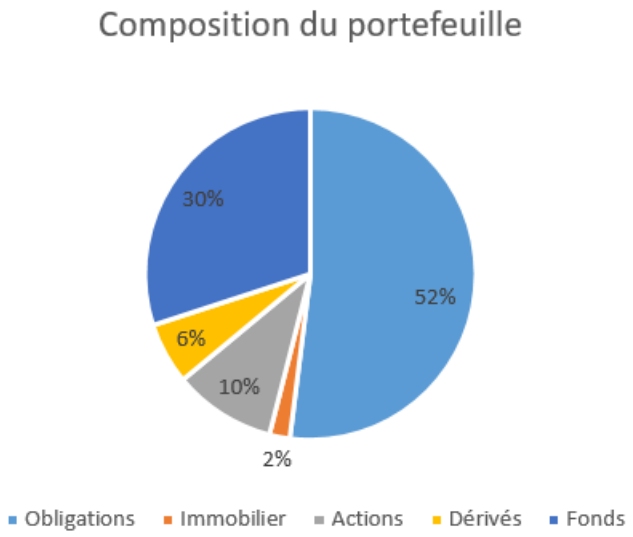



FIGURE 6 – Composition du portefeuille d'actifs

On peut voir que la majorité des actifs sont des obligations avec 58% ce qui est cohérent avec la plupart des assureurs sur le marché. Les seconds types d'actifs majoritaires sont les fonds avec 30%, suivis par les actions 10%, les dérivés à 6% et l'immobilier qui vient compléter le tout avec

ses 2%.

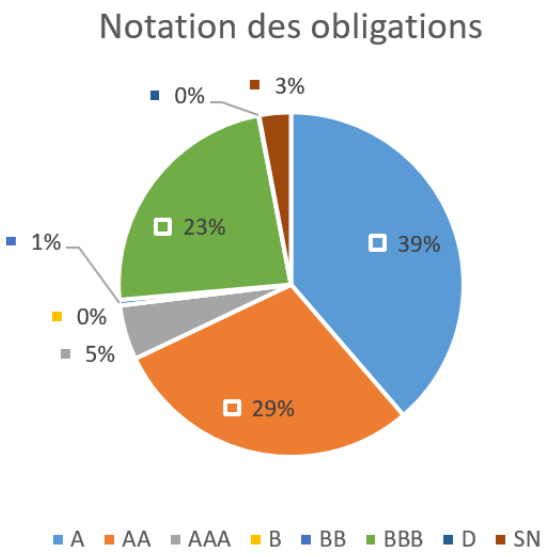



FIGURE 7 – Notation des obligations

En venant étudier plus en détail le type d'actif majoritaire, nous pouvons constater qu'il est majoritairement constitué de produits avec une notation supérieure ou égale à A (73% de l'ensemble des obligations), suivis par 23% notés BBB. Les produits obligataires sont donc majoritairement constitués de titres de qualité supérieure et de haute qualité (resp. A et AA), avec une part non négligeable de titres de qualité moyenne (BBB).

### 2.1.2 Description du portefeuille d'assurés

Les données du portefeuille d'assurés ne seront pas directement utilisées. En effet, la variation des données au passif est bien trop faible (manque de sensibilités, changement faible dans une même année) pour que cela soit facilement utilisable par nos modèles de machine learning. En effet, sur les centaines de milliers de lignes que la base de données finale comprendra, il y aura au plus une dizaine de portefeuilles de passif différents ce qui est totalement négligeable. Nous capterons quand même certaines quantités et variables liées au passif telles que la provision mathématique ou encore le taux de rachat ou de décès ce qui nous permettra de ne pas ignorer complètement les données du portefeuille.

Il est tout de même intéressant d'étudier ce dernier, étant donné que d'autres variables seront directement influencées par les caractéristiques du portefeuille.

Les caractéristiques du portefeuille de passif sont alors les suivantes :

— La moyenne d'âge est de **72 ans** la médiane à **73 ans**. Ces indicateurs étant assez élevés, nous pouvons nous attendre à des prestations décès qui seront importantes.

— L'ancienneté moyenne est de **13.2 ans**

— L'âge à la souscription est en moyenne de **59** ans.

— Le portefeuille est composé de **4 445 460 contrats**.

— Les provisions mathématiques totales sont de **96 343 197 650.54€**, avec une moyenne de **21 672.26€** par contrat.

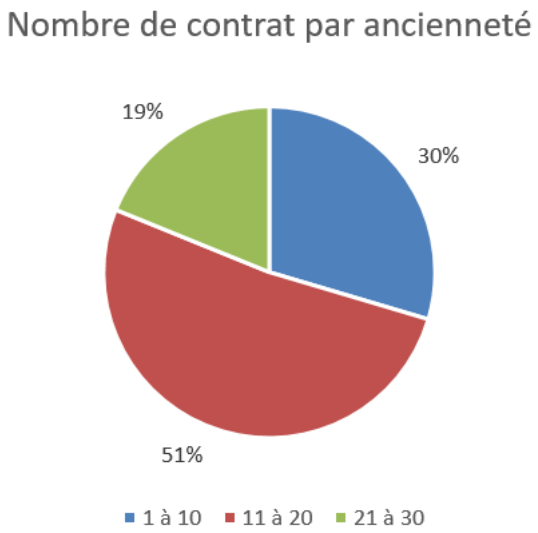



FIGURE 8 – Nombre de contrats par groupe d'ancienneté

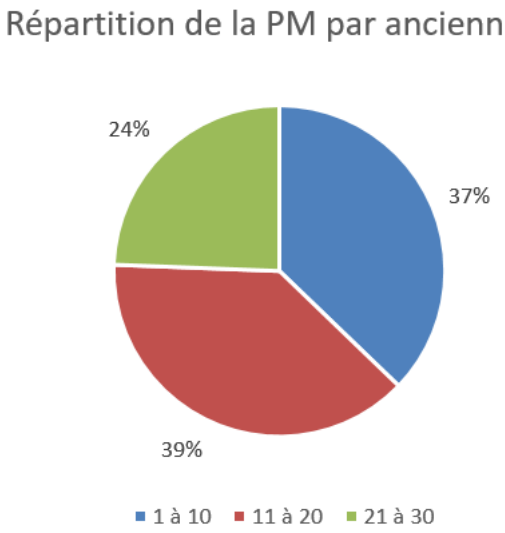



FIGURE 9 – Provisions mathématiques par groupe d'ancienneté

On constate que la majorité des contrats ont une ancienneté entre 11 à 20 ans avec 51% des contrats dans ce groupe, tandis que les contrats les plus anciens sont minoritaires avec seulement 19% des contrats ayant entre 21 et 30 ans d'ancienneté. Les contrats récents viennent quant à eux compléter le tout avec 30% des contrats. Cependant, lorsque l'on analyse la répartition des groupes d'ancienneté en prenant en compte les provisions mathématiques sous jacentes, on observe que la répartition de la PM est plus équilibrée comparée au nombre de contrats. On peut en déduire que les contrats d'ancienneté moyenne sont ceux qui ont la prime pure la plus faible. Cela peut potentiellement s'expliquer par le fait que les contrats anciens sont rattachés à des assurés plus vieux et donc avec un risque de décès plus important, tan-

dis que les contrats les plus jeunes sont ceux liés à une population, elle aussi, jeune et avec un profil plus risqué.

## 2.2 Création de la base de données

Afin de pouvoir créer notre base de données, nous aurons besoin de décider de quoi récupérer et de comment transformer l'ensemble de nos données afin d'obtenir une base finale suffisamment dense pour nos modèles. De plus, il est nécessaire que cette base soit dans un format exploitable et compréhensible et avec le moins de données superflues possibles. Dans le cas d'une régression, le problème peut se résumer à :

Étant donné $N$ variables explicatives $Y_i$, comment estimer et prédire la variable cible $X$.

La structure la plus évidente est donc celle d'un tableau en deux dimensions contenant $N+1$ colonnes et $M$ lignes. $M$ correspond alors au nombre de variations (scénarios, stochastiques ou non) de nos données dont le résultat est connue grâce à l'oracle (ici NEMO).

TABLE 2 – Format typique de la base de données finale souhaitée

| | $Y_1$ | $Y_2$ | $\dots$ | $Y_N$ | $X$ |
|---|---|---|---|---|---|
| 1 | $y_{(1,1)}$ | $y_{(1,2)}$ | $\dots$ | $y_{(1,N)}$ | $x_1$ |
| 2 | $y_{(2,1)}$ | $y_{(2,2)}$ | $\dots$ | $y_{(2,N)}$ | $x_2$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\dots$ | $\vdots$ | $\vdots$ |
| $M$ | $y_{(M,1)}$ | $y_{(M,2)}$ | $\dots$ | $y_{(M,N)}$ | $x_M$ |

Grâce au processus de production de CNP Assurances, nous avons à disposition des données structurées de bonne qualité qui ont été certifiées par des experts métiers qui effectuent des contrôles réguliers avant d'être transmis aux régulateurs. Cependant, elles n'ont pas forcément les mêmes dimensions et variables. En effet, les variables des ESG et des sorties d'indicateursAvecPB sont toutes les deux des tableaux de taille $50 \times 1000$ (50 pas de temps et 1000 scénarios dans le cas stochastique), les données de MPActif ne sont pas projetées dans le temps et ne sont pas stochastiques.

### 2.2.1 Traitement et uniformisation des variables

Afin de prédire le SCR de l'entreprise, plusieurs solutions s'offrent à nous. En effet, en fonction de la maille à laquelle nous prédisons nos variables, nous pouvons obtenir des résultats différents. La méthode la plus simple serait de prédire directement le SCR global de l'entreprise. Mais il serait aussi possible de prédire les SCR modulaires puis de les réagreger, ou de prédire les SCR sous-modulaires et de faire de même. Le fait de prédire sur une maille plus fine nécessite aussi d'avoir un modèle pour chaque composante prédite. Avec un nombre important de composantes, le temps nécessaire à l'entraînement et l'optimisation des modèles peut remettre en cause la pertinence de l'utilisation du machine learning en premier lieu qui visait à avoir des résultats de manière rapide.

Plus la prédiction arrive à une maille fine, moins il y aura d'erreur sur la suite de l'agrégation. Cependant, cela pose la question d'une démultiplication de l'erreur de prédiction. Une prédic-

tion à une maille très fine n'est pertinente que si cette prédiction est suffisamment précise pour que l'agrégation de ces prédictions ne résulte pas en une erreur incontrôlée.

Plusieurs choix sont possibles lors de la création de la base de données. Il est possible de récupérer plus de données et les avoir à une maille plus fine ce qui peut permettre d'avoir une plus grande précision et potentiellement amener à un modèle plus performant, mais l'augmentation du nombre de variables peut être problématique pour l'entraînement de nos modèles. En effet, l'entraînement et l'optimisation des modèles demandent énormément de ressources. Garder des variables inutiles ou à une maille trop fine risque de nous pénaliser en nous empêchant d'optimiser correctement nos modèles. A l'inverse, trop peu de variables peu nous empêcher de capter des subtilités dans la structure de nos données et donc donner un modèle qui sera trop grossier.

Plusieurs étapes seront donc nécessaires à la création de notre base de données :

— **Récupération des données**

— **Traitement des données**

— **Transformation des formats**

— **Suppression des variables non pertinentes**

### 2.2.2 Récupération des données

La récupération des données n'était pas une tâche complexe étant donné que les entrées et sorties des modèles de CNP Assurances sont déjà structurées et disponibles pour la plus grande partie. Dans le cadre du mémoire, nous avons récupéré les principales sensibilités effectuées sur un portefeuille en euros de CNP Assurances. Nous avons de plus uniquement considéré les données du T4 2021. Ces choix ont été effectués afin de limiter le temps passé à la récupération des données qui, bien que disponibles, doivent être récupérées à la main. De plus, comme dit précédemment, une trop grosse base de données ralentirait considérablement la vitesse à laquelle nous pouvons entraîner, optimiser et valider nos modèles. Il n'est cependant pas exclu d'ajouter de nouvelles sensibilités, portefeuilles ou arrêtés afin de compléter la base de données par la suite.

Nous avons donc récupéré les données suivantes :

— 1 arrêté : T4 2021

— 40 sensibilités différentes (sensibilité SII ainsi que Bâle III)

— Une version déterministe et une version stochastique par sensibilité

— 22 fichiers par version

Tout cela pour un total de **1760** fichiers avec une taille finale de plus de **100Go** de données.

### 2.2.3 Traitement des données

Les données utilisées étant les données officielles de CNP Assurances, il n'y a pas de biais sur la population choisie ou les données utilisées pour notre modèle. Ces données, bien que de très grande qualité, contiennent des informations qui ne sont pas forcément pertinentes pour

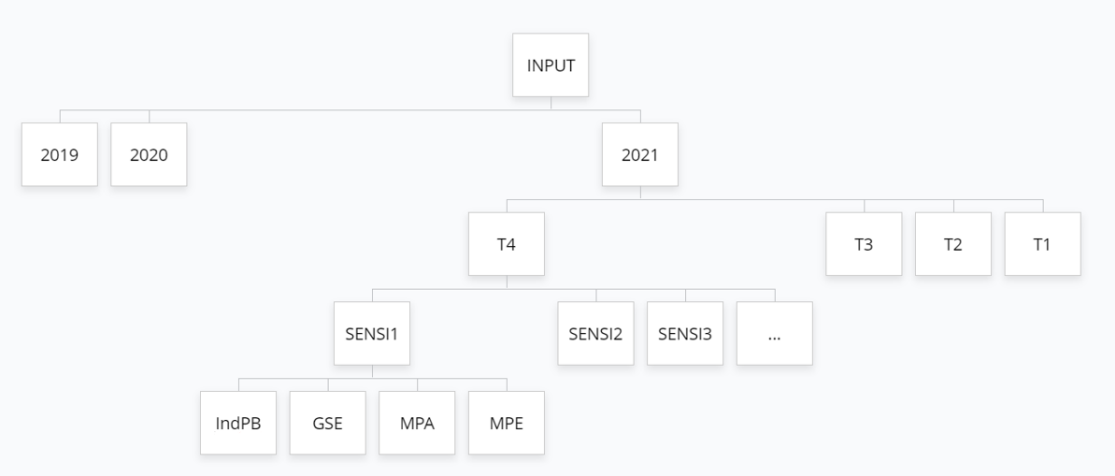



FIGURE 10 – Arborescence de stockage des données

nos modèles, ou alors ont besoin d'être retraitées pour éliminer certaines valeurs qui seraient aberrantes ou provoqueraient des erreurs pour nos algorithmes.

Parmi ces variables sont comprises celles qui n'ont pas de valeur (les NA), celles qui contiennent des "flags" (valeurs indicateurs n'ayant pas de sens d'un point de vue numérique), et les variables non quantitatives.

Les premières sont généralement non assignées car non pertinentes ou alors correspondent à un cas spécifique pour un portefeuille tiers. Pour régler cela, nous avons découpé les variables NA en deux catégories :

— Les variables complètement NA (aucune valeur)

— Les variables contenant des valeurs NA

Pour les premières, nous avons pris la décision de simplement supprimer la variable tandis que pour la seconde, nous avons décidé d'au lieu de retirer la ligne correspondante de notre base de données, de la garder afin de pouvoir exploiter les autres données non NA mais de remplacer cette valeur par la moyenne de cette variable qui n'est pas NA.

De manière évidente, cela ne s'applique que pour les variables quantitatives, les variables qualitatives étant traitées différemment.

Après ce premier traitement pour ne garder que des valeurs adéquates, nous avons besoin d'effectuer du feature engineering, c'est à dire retraiter et transformer les données et variables afin de les rendre utilisables comme données d'entraînement par nos modèles.

Une des modifications de variable à effectuer concerne les variables qualitatives qui doivent être transformées en variables quantitatives. En effet, la plupart des modèles ne travaillent qu'avec des données quantitatives et il nous est nécessaire de transformer ces variables par des procédés pertinents pour la prédiction de notre résultat. Cependant, certaines de ces variables contiennent un très grand nombre de modalités (par exemple le type d'actifs dans le MPA), il nous faut donc choisir sagement quelles modalités nous considérons comme étant les plus importantes et comment nous voulons les garder/représenter.

Pour cela, une étude variable par variable a été effectuée pour étudier les variables qualitatives et déterminer quelles sont les modalités jugées intéressantes. Ces modalités retenues seront exprimées en pourcentage par rapport à la PM ou à la VM totale représentée.

Pour le cas d'une variable liée à la PM, cela donnerait :

$$Pourc_{Mod} = \sum_{i \in I} \frac{PM_i}{PM_{tot}} \tag{5}$$

avec $I$ l'ensemble des lignes de la base de données considérée contenant la modalité $Mod$.

Pour les codes S2 (code spécifique à CNP Assurances permettant d'associer une classe d'actif à un choc S2 spécifique) du model point d'actif, nous passons alors de ces modalités :

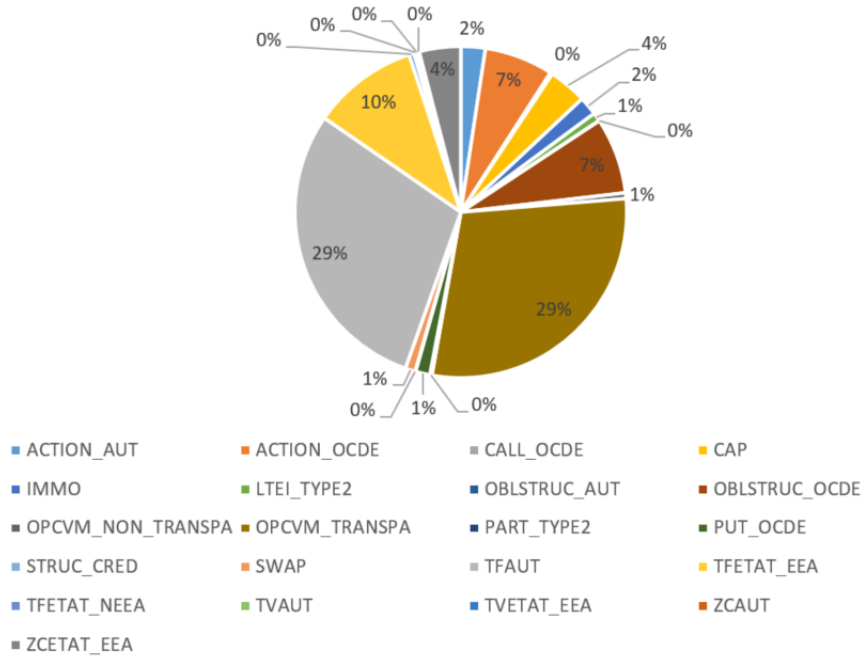



FIGURE 11 – Modalités avant retraitement

aux modalités suivantes :

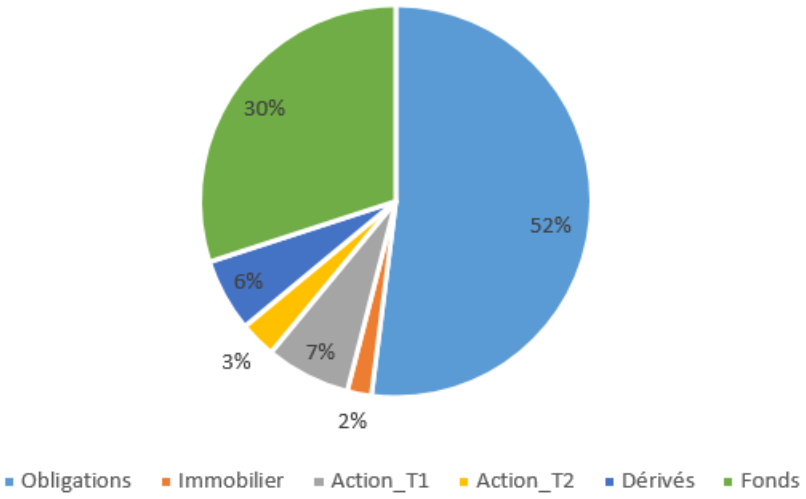



FIGURE 12 – Modalités après retraitement

Cette opération est alors effectuée pour l'ensemble des variables qualitatives et l'ensemble des nouvelles modalités sont déterminées par avis d'expert.

### 2.2.4 Transformation des formats

Après avoir été retraitées, il est alors nécessaire de transformer nos variables pour les obtenir dans les formats attendus. En effet, nous souhaitons obtenir une ligne par scénario et chaque scénario contient des variables pouvant être dans des formats très variés. Il a donc été nécessaire de trouver des statistiques adaptées afin de représenter de la meilleure manière possible nos différents variables sur une unique ligne de tableau. Dans le cas des variables qualitatives, la transformation était relativement simple. Nous avons décidé de garder les cinq modalités les plus importantes proportionnellement à la valeur boursière ou à la provision mathématique et nous avons alors une nouvelle variable par modalités gardées. Dans l'exemple du code S2 vu précédemment, les modalités retenues et donc les nouvelles variables sont :

— Obligations
— Fonds
— Action_T1
— Dérivés
— Immobilier

La modalité **Action_T2** n'a pas été retenue suite à cette transformation.

Concernant les variables quantitatives, une démarche plus simple et systématique a été retenue. Pour les variables du model point d'actif, le choix de la statistique s'est porté sur une moyenne pondérée par la valeur boursière de l'actif considéré comme montré dans la dernière équation. Nous n'avons pas considéré d'autres statistiques car la dispersion de nos variables est assez faible et nous ne voulions pas surcharger la base de données avec des variables inutiles.

Une première version avait néanmoins été considérée avec des indicateurs supplémentaires mais elle a été abandonnée car cette augmentation d'indicateurs alourdissait considérablement nos modèles.

Pour les variables liées au générateur de scénarios économiques, nous avons décidé d'utiliser plus d'indicateurs que pour le MPA. En effet, les indicateurs du GSE sont ,en plus d'être stochastiques, projetés dans le temps sur 50 ans. Nous avons alors décidé de garder les statistiques suivantes :

— Moyenne

— Variance

— Quantile à 25%

— Quantile à 75%

— Quantile à 99%

En plus de ces statistiques, il était nécessaire de choisir quelles périodes nous voulions retenir. Dans l'optique d'effectuer le moins de choix pouvant potentiellement biaiser nos résultats, nous avons décidé d'effectuer ces statistiques sur l'ensemble des 50 années de projection. Il aurait néanmoins pu être intéressant de ne garder que les premières années, qui vont avoir un impact plus direct sur nos résultats, ou de séparer par décennie afin de garder l'information sur l'ensemble de la période de projection.

Les variables des sorties de NEMO(IndicateursAvecPB) sont quant à elles traitées différemment car elles suivent une logique différente des variables de l'ESG. En effet, bien que les variables soient aussi projetées dans les temps, nous disposons en plus des taux d'actualisation liés à la variable. Une moyenne aurait donc peu de sens car nous ne prendrions pas en compte l'actualisation de nos valeurs. C'est donc pour cela que vous avons décidé de récupérer la **Valeur Nette Actualisée** pour représenter nos variables de sortie dans notre base de données.

Soit $x_i$ la variable et $f_i$ le facteur d'actualisation en question pour la date $i$ (en années) et $I$ l'ensemble des années projetées.

La valeur stockée sera donc :

$$VAN_x = \sum_{i \in I} x_i \times f_i \tag{6}$$

Cependant, une décision a été prise de ne pas utiliser certaines variables de IndicateurAvecPB telles quelles. En effet, les données de cette sortie sont telles qu'une formule fermée existe pour retrouver la valeur de l'Equity et avoir ces valeurs dans notre base de données serait totalement contre-productif. Le modèle retrouverait assez facilement la relation qui est linéaire et le meilleur modèle aurait une erreur extrêmement faible. De plus, le modèle serait très difficilement utilisable en gardant ces variables brutes : Les variables retenues deviendront des entrées de notre modèle afin d'en prédire l'Equity. Garder ces variables reviendrait à les demander en entrée et nous serions incapables de fournir des approximations de ces données avec une telle précision.

Afin de trouver un compromis et de ne pas complètement perdre les informations sur ces variables, nous avons décidé de les représenter sous forme de pourcentages.

Soit $X$ une variable que nous voulons transformer en pourcentage. Cette variable est projetée sur 50 années et (dans le cas stochastique) dispose de 1000 variations différentes dues aux différents scénarios. Ces variables sont soit liées à la valeur boursière (VB), soit à l'encours moyen passif (EMP), bien que l'écart entre les deux soit relativement faible. Les valeurs sont alors transformées en gardant la valeur proportionnellement à la VB ou à l'EMP et la moyenne sur les 10 premières années est stockée dans la base de données.

Soit $X$ notre variable, dépendant ici de la valeur boursière totale du portefeuille, alors pour obtenir sa nouvelle valeur, le calcul suivant sera effectué :

$$\overline{X}_{nY} = \sum_{i=1}^{n} \frac{x_i \times f_i}{VBOURS_i}$$

avec :

— $\overline{x}_{nY}$ le taux ou pourcentage moyen sur 10 ans.

— $x_i$ la valeur de la variable pour l'année $i$.

— $f_i$ le facteur d'actualisation de l'année $i$.

— $VBOURS_i$ la valeur boursière du portefeuille pour l'année $i$.

Et pour finir, certaines variables donnant des informations qualitatives non transformables car non stochastiques ont été stockées par la méthode du one hot encoding. Ces variables sont notamment les sensibilités utilisées ou encore l'arrêté considéré.

**One-hot-encoding**

Le one hot encoding est une méthode d'encodage largement utilisée en machine learning permettant de transformer des variables qualitatives avec un certain nombre de modalités en variables quantitatives. Pour cela, chaque modalité de la variable de départ est transformée en variable quantitative ne pouvant prendre que deux valeurs : 1 (Vrai) ou 0 (Faux).

Cette représentation imite les circuits électroniques dans lesquels un switch peut être ouvert ou fermé.

Avec cette représentation, nous pouvons alors facilement représenter ces variables qualitatives en variables quantitatives qui seront aisément interprétées par la majorité des modèles de machine learning. Il est cependant aussi assez facile de voir les désavantages qu'elle apporte : Dans le cas où la variable à transformer dispose de nombreuses modalités, le nombre de variables de notre base de données exploserait rapidement.

Exemple avec une variable $X$ contenant 4 modalités $(A, B, C, D)$

### 2.2.5 Suppression des variables non pertinentes

La suppression des variables non pertinentes est une étape importante dans la construction de la base de données. Elle permet en effet de réduire la taille des données sur laquelle nous allons

| | $X$ |
|---|---|
| 1 | A |
| 2 | B |
| 3 | B |
| 4 | C |

↓

| | $X_A$ | $X_B$ | $X_C$ | $X_D$ |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 |

TABLE 3 – Illustration du one hot encoding

travailler et donc améliorer les temps d'entraînement et d'optimisation. De plus, cela permet de retirer des informations redondantes qui risquent de bruiter le modèle et donc de ne pas fournir le meilleur résultat possible.

Pour cela, nous avons procédé en deux étapes :

— Réduction de variables par avis d'expert

— Réduction de variables par algorithmes

La réduction de variables par avis d'expert a été effectuée de pair avec un expert du modèle NEMO utilisé par CNP Assurances ce qui nous a permis d'identifier les variables n'étant pas intéressant dont la présence dans les entrées/sorties était due à des fonctions annexes, les variables redondantes qui donnaient des informations très semblables avec comme seules différences les régulations desquelles elles sont issues, ou encore des variables donnant des informations à une maille beaucoup trop fine pour notre usage.

Après cette première sélection, nous passons alors de 908 variables pour la première version sans sélection de variables par avis d'expert, à 143 variables lors de la deuxième version.

Ces variables vont alors servir de socle de base pour appliquer nos premiers modèles de machine learning. La sélection de variables par algorithme ne sera effectuée qu'à une étape ultérieure, cet algorithme pouvant éliminer potentiellement un très grand nombre de variables, nous souhaitons conserver le maximum de variables lors des premiers tests et optimisations.

### 2.2.6 Base de données finale et analyse des données

Après l'ensemble de ces étapes, nous obtenons alors notre base de données finale qui est composée de **720720** lignes et **143** colonnes.

Notons que dans notre base de données, le one hot encoding a été fait avec des booléens (True/False) car leur représentation sous Python est équivalente aux entiers 1 et 0.

Nous pouvons désormais explorer nos variables créées et regarder comment elles interagissent entre elles, ou comment elles sont réparties selon nos scénarios. De manière évidente, nous n'allons pas pouvoir analyser l'intégralité des variables étant donné leur nombre important

| | ALLOCATION_STRATEGIQUE_Diversification | ALLOCATION_STRATEGIQUE_Taux | ... | ChocSpreadObligation | ChocSpreadTitrisation |
|---|---|---|---|---|---|
| 0 | 0.222299 | 0.733834 | ... | False | False |
| 1 | 0.222299 | 0.733834 | ... | False | False |
| 2 | 0.222299 | 0.733834 | ... | False | False |
| 3 | 0.222299 | 0.733834 | ... | False | False |
| 4 | 0.222299 | 0.733834 | ... | False | False |
| ... | ... | ... | ... | ... | ... |
| 720715 | 0.222299 | 0.733834 | ... | False | False |
| 720716 | 0.222299 | 0.733834 | ... | False | False |
| 720717 | 0.222299 | 0.733834 | ... | False | False |
| 720718 | 0.222299 | 0.733834 | ... | False | False |
| 720719 | 0.222299 | 0.733834 | ... | False | False |

720720 rows × 143 columns

FIGURE 13 – Extrait de la base de données finale

mais nous allons plutôt nous concentrer sur la variable **Equity** qui est notre variable à prédire.

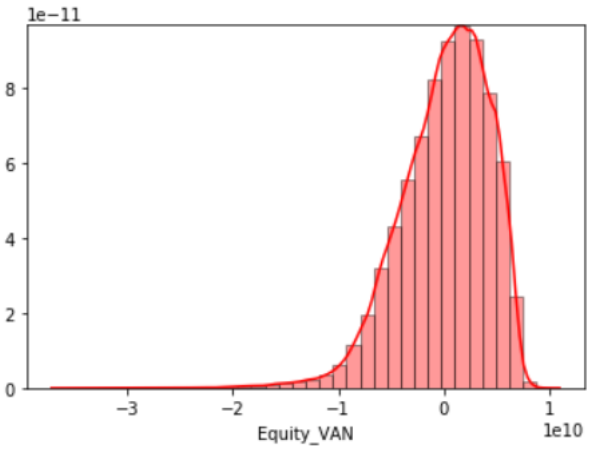



FIGURE 14 – Analyse univariée de l'Equity

Nous pouvons voir que la distribution de l'Equity ne suit pas tout à fait une loi normale, une asymétrie étant visible. Cela pourrait faire penser à une Gamma en inversant l'axe des abscisses et en normalisant les valeurs. L'étude de la distribution de la variable cible sera effectuée dans une partie différente.

Une étude bivariée de la variable cible avec certaines variables explicatives nous permet d'avoir une intuition sur comment la première va réagir en fonction de nos différents scénarios.

L'interprétation de ces graphiques se fait de la manière suivante :

— Des points concentrés sur une ligne horizontale indique une faible variation de la variable explicative considérée.

— Une concentration sur une ligne verticale indique quant à elle un faible impact de cette variable explicative sur la variable à prédire (les variations de cette première n'affectent pas les résultats de la seconde).

— Des points concentrés dans le quadrant supérieur droit ou inférieur gauche indique une

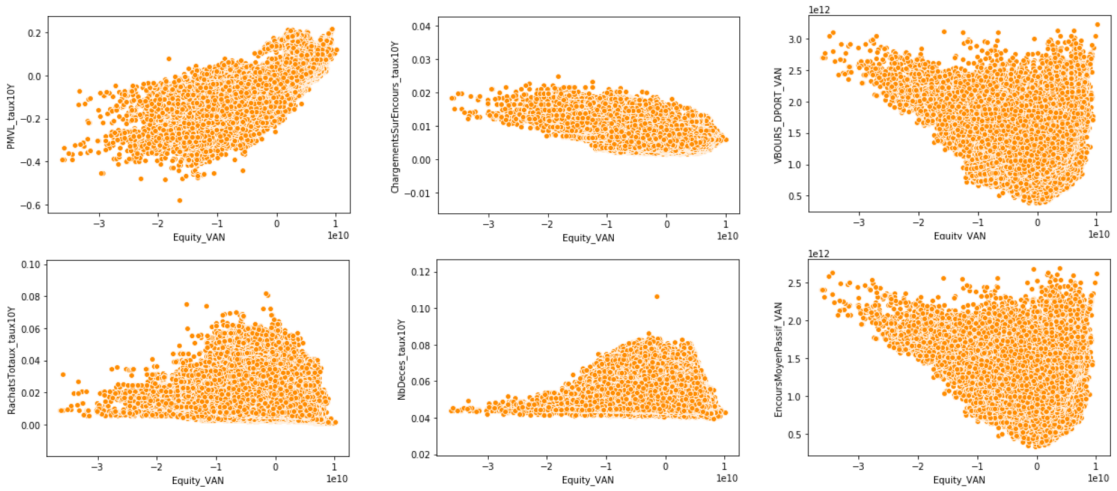



FIGURE 15 – Analyses bivariées de l'Equity avec (de gauche à droite, de haut en bas) : *PMVL_taux10Y, ChargementSurEncours_taux10Y, VBOURS_VAN, Rachatstotaux_taux10Y, NbDeces_taux10Y, EncoursMoyenPassif_VAN.*

corrélation positive sur ces intervalles : une forte (ou faible) valeur dans la variable explicative sera alors souvent accompagnée d'une forte (resp faible) valeur pour la variable à prédire.

— Dans le cas où la concentration se fait dans les quadrants supérieurs gauche ou inférieur droit, la corrélation est alors négative et on a un effet opposé entre la variable explicative et la variable à prédire.

Dans les variables choisies, on peut observer une corrélation positive entre l'Equity et la PMVL, ce résultat est attendu étant donné que ce dernier constitue une part non négligeable de l'Equity.

Les analyses bivariées pour la valeur boursière et l'encours moyen passif sont très similaires. En effet, dans le cas d'un bilan équilibré le passif est égal à l'actif et on a donc Encours Moyen Passif = Valeur Boursière. C'est d'ailleurs un test de cohérence courant des bilans.

Dans les cas des variables restantes, entre autres les chargements sur encours ou les rachats totaux, les liens avec l'Equity est plus difficile à quantifier et à observer. Ces variables n'agissent pas de manière explicite sur l'Equity et donc leurs effets sur cette dernière sont moins appréciables et leurs impacts plus difficilement estimables.

Cette matrice de corrélation est calculée à partir de la corrélation de Pearson. Ce coefficient est une mesure de la corrélation **linéaire** entre deux jeux de données : ici, deux variables de notre base de données.

Soient $X$ et $Y$ deux variables de notre base de données, et $x_i$ et $y_i, i \in \mathbf{I}$ leurs valeurs pour chaque scénario $i$.

On a alors :

$$\rho_{X,Y} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

Avec :

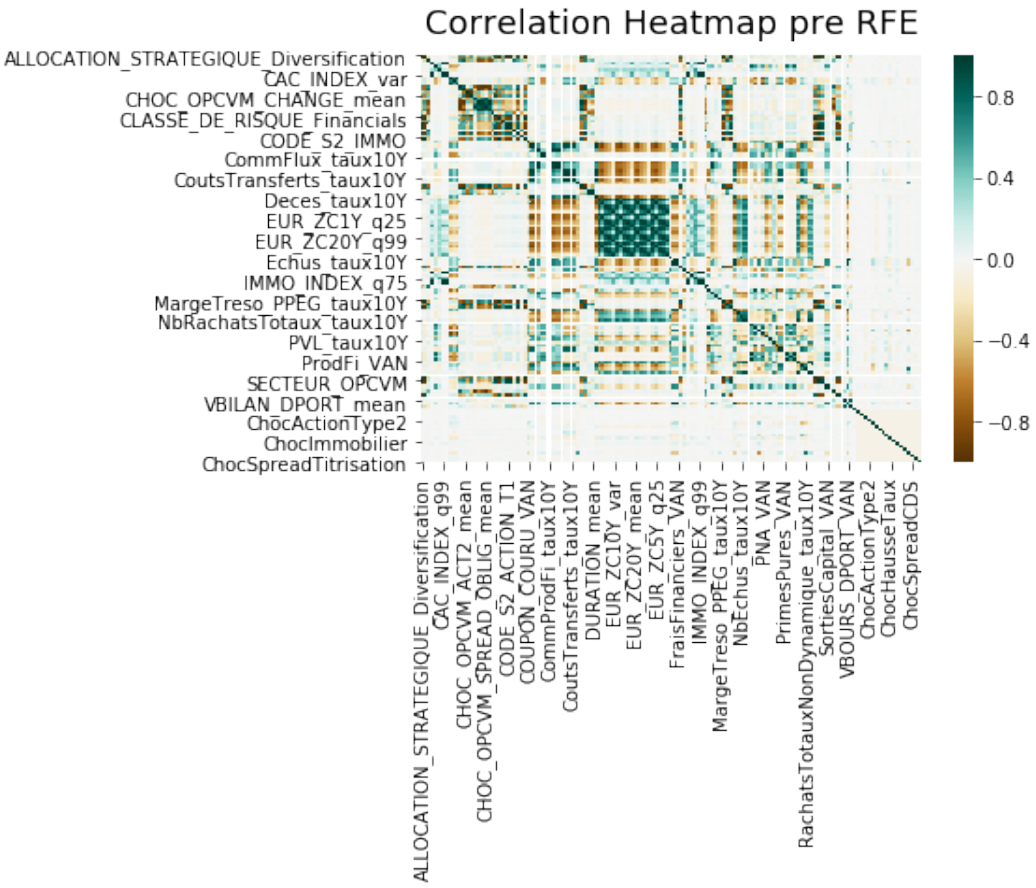



FIGURE 16 – Corrélation entre les variables de la base de données avant toute réduction de variables

— $\sigma_X$ l'écart-type de la variable X

— $\sigma_Y$ l'écart-type de la variable Y

— $Cov$ la fonction de covariance

En considérant les indicateurs empiriques suivant :

— $\overline{x} = \sum_{i \in \mathbf{I}} \frac{x_i}{|\mathbf{I}|}$

— $\overline{y} = \sum_{i \in \mathbf{I}} \frac{y_i}{|\mathbf{I}|}$

— $|\mathbf{I}|$ étant le nombre d'éléments de $\mathbf{I}$

On a alors :

$$\rho_{X,Y} = \frac{\sum_{i \in \mathbf{I}} (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i \in \mathbf{I}} (x_i - \overline{x})^2} \sqrt{\sum_{i \in \mathbf{I}} (y_i - \overline{y})^2}} \tag{7}$$

En utilisant une mesure de corrélation différente comme celle de Kendall, une matrice de corrélation assez similaire peut être obtenue. Le coefficient de Kendall est une mesure de corrélation non linéaire basée sur les rangs. La similarité entre les deux matrices de corrélation indique que les dépendances entre nos différentes variables sont principalement linéaires.

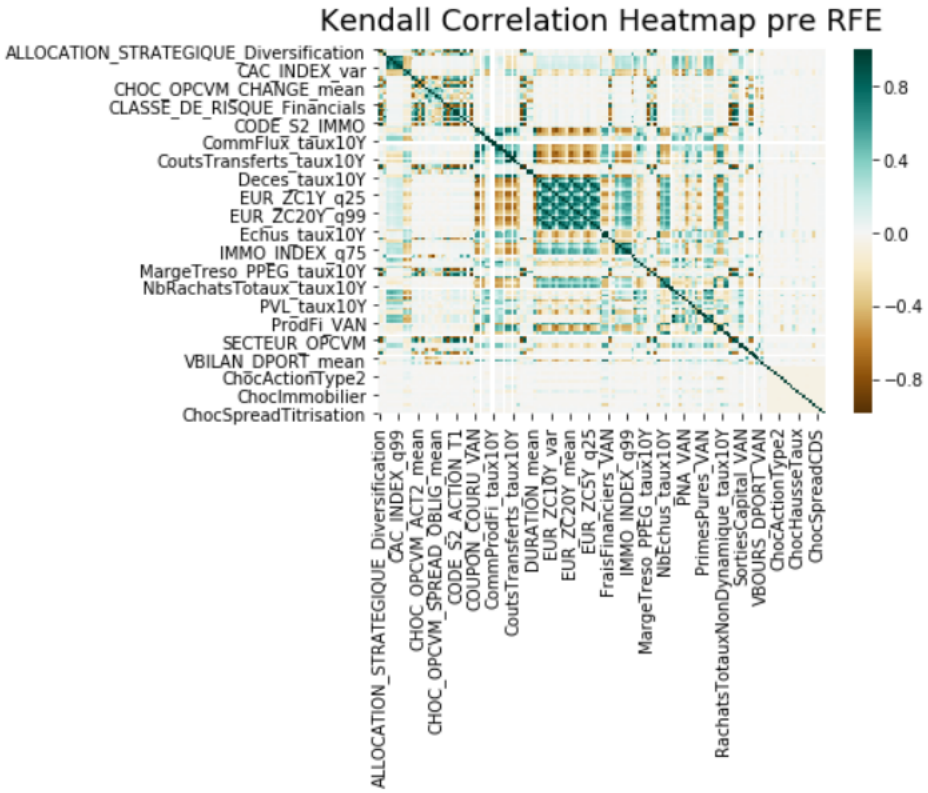



FIGURE 17 – Matrice de corrélation de Kendall avant réduction des variables

Le tau de Kendall se calcule en comparant le nombre de paires concordantes avec le nombre de paires discordantes.

**Définition** (Paires concordantes et discordantes)**.** *Soient $X$ et $Y$ deux variables et $x_i, x_j, y_i$ et $y_j$ des observations de ces variables. Les paires $(x_i, y_i)$ et $(x_j, y_j)$ sont dites concordantes si la propriété $x_i > x_j$ et $y_i > y_j$, ou $x_i < x_j$ et $y_i < y_j$ est vérifiée. Dans le cas contraire, les paires seront discordantes*

**Définition** (Tau de Kendall)**.** *Le tau de Kendall est alors défini de la manière suivante :*

$$\tau = \frac{(\textit{nombre de paires concordantes}) - (\textit{nombre de paires discordantes})}{(\textit{nombre de paires})} \tag{8}$$

$$= 1 - \frac{2 \times (\textit{nombre de paires discordantes})}{\binom{n}{2}} \tag{9}$$

La matrice de corrélation permet de mettre en lumière la corrélation entre les différentes variables explicatives de notre base de données. En théorie, il serait souhaitable d'avoir des variables complètement non corrélées. En effet, dans cette situation, les variables n'amèneraient alors pas d'informations déjà présentes via d'autres variables et, pour peu que les variables considérées aient un impact sur l'Equity, il n'y aurait aucune information redondante et donc la base de données serait optimisée.

Il est évident que dans un cas réel, cette situation ne se produit quasiment jamais. Nos variables provenant de modèles faisant appel à de nombreuses structures de données et d'indicateurs

différents, des variables sont destinées à être plus ou moins corrélées.

Il existe cependant diverses techniques pour réduire la corrélation inter-variables, notamment la technique du Principal Component Analysis. Cette technique ne sera pas explorée dans ce mémoire car sa mise en place lorsque des variables qualitatives sont présentes est assez lourde, mais peut être intéressante à intégrer par la suite. L'algorithme sous-jacent va tenter de trouver la combinaison linéaire de nos variables avec le plus fort pouvoir explicatif possible et construire de nouvelles variables à partir de ces dernières, jusqu'à obtenir un nouveau jeu de variables qui sera alors moins corrélé.

# 3 Application des modèles de Machine Learning et réduction des dimensions

## 3.1 Introduction au Machine Learning ou apprentissage automatique

Nous allons ici effectuer un résumé de ce qu'est le machine learning, ses origines, utilisations, avantages et faiblesses avant de nous attaquer à son application dans le cadre de ce mémoire en commençant par la création de la base de données qui va servir à la fois de base d'apprentissage ainsi que de base de validation pour nos différents modèles.

Le machine learning (ML ou apprentissage automatique en français) est un domaine de l'intelligence artificielle, au croisement entre mathématiques et informatique. Cette discipline se base sur les mathématiques et les statistiques afin de pouvoir faire "apprendre" aux ordinateurs à partir d'une base de données, et donc de résoudre des problèmes et effectuer des tâches sans forcément leur indiquer les démarches à suivre explicitement. Au lieu de suivre un algorithme classique dont l'ensemble des actions auront été explicitement décrit par un humain, les modèles de machine learning vont optimiser des fonctions de manière autonome jusqu'à obtenir un résultat jugé optimal en fonction des hypothèses et paramètres donnés.

L'idée du machine learning a été principalement développée par Alan Turing via son concept de "machine universelle" en 1936 et posera les bases de l'apprentissage automatique. De nombreuses avancées ont été effectuées dans les années qui ont suivi et le terme de machine learning viendra au jour en 1959 par l'informaticien Arthur Samuel qui développera un programme pouvant jouer au jeu de Dames et s'améliorant au fil des parties. Le véritable potentiel du machine learning ne commencera à être exploité qu'à partir de 1997 où le programme Deep Blue développé par IBM parviendra à vaincre le champion mondial d'échec Garry Kasparov. Aujourd'hui, il existe de nombreux programmes faisant usage du machine learning pour des tâches très variées, allant de la reconnaissance de visages au jeu de Go considéré comme le jeu de plateau le plus dur au monde.

En 2014, un programme parvient même à passer le test de Turing, test qui consiste à tromper au moins 30% des juges humains durant un test de 5 minutes en se faisant passer pour non pas un programme informatique mais pour une personne réelle, donnant alors un réel poids au terme d'intelligence artificielle.

Les méthodes de machines learning et ses applications sont très diverses, il est néanmoins possible de les regrouper en plusieurs catégories qui correspondent à différentes manières avec lesquelles le programme va "apprendre" :

— **L'apprentissage non supervisé** : Avec cette méthode d'apprentissage, notre programme ne dispose que d'exemples mais pas d'étiquettes. Il n'a donc pas d'informations concernant le nombre de classes et on va alors parler de *clustering*. L'algorithme va essayer de trouver de lui-même les liens entre les données et les structures qui les composent.

— **L'apprentissage supervisé** : Contrairement à l'apprentissage non supervisé, le programme dispose ici des étiquettes des données et va donc essayer de classer les données dans

les étiquettes existantes. Ce type d'apprentissage nécessite un expert ou ( *oracle*) qui va devoir étiqueter correctement les données d'entraînement au préalable. Dans le cas où nous disposons d'un nombre de classe infini dans un domaine continue (ie la grandeur à classer est continue), on va alors parler de régression.

— **L'apprentissage par renforcement** : Dans ce contexte, le programme va effectuer des actions en passant par un agent autonome qui vont influer sur son environnement. Ce dernier va alors réagir aux décisions de l'agent et lui procurer une récompense positive ou négative. L'agent adapte ensuite ses actions dans le but d'optimiser la récompense totale obtenue.

Afin de pouvoir apprendre correctement des données fournies, plusieurs conditions sont souvent nécessaires sur la base de données afin d'obtenir un programme fournissant un résultat fiable.

— **Volume de donnée important** : Afin de pouvoir détecter des structures plus complexes dans les données ainsi que de réduire les mauvais choix par l'algorithme, la base de données doit être suffisamment abondante. La taille de la base de données est fortement liée à la complexité du problème et il n'existe donc pas de nombre de donnée minimale afin de pouvoir utiliser ces algorithmes.

— **Diversité des données** : En plus de nécessiter un grand nombre de données, ces dernières doivent être suffisamment diversifiées pour que l'ensemble des situations possibles soient représentées dans la base de données. Une sous diversité des données risque de biaiser le modèle en favorisant les classes (dans le cas d'un apprentissage supervisé) les plus courantes. Pour la même raison, il est important que les scénarios plus rares soient quand même représentés en nombre suffisant pour ne pas qu'ils soient ignorés.

Cependant, ces deux conditions vont mener à des temps d'entraînement et d'optimisation pour nos modèles pouvant être très longs. Il faut donc faire attention de ne garder que les informations pertinentes pour le modèle ou trouver des méthodes pour contourner ces potentielles limitations.

## 3.2 Présentation des différents modèles

Parmi les différents modèles de machine learning existant, les plus adaptés pour le problème considéré sont naturellement les algorithmes de régression supervisés. Pour cela les modèles suivants seront intéressants pour répondre à la problématique de ce mémoire :

— Le modèle Random Forest

— Le modèle eXtreme Gradient Boosting

— Le modèle Support Vector Regression

Les arbres de classification et de régression seront aussi introduits, étant une brique de base pour nos deux premiers modèles.

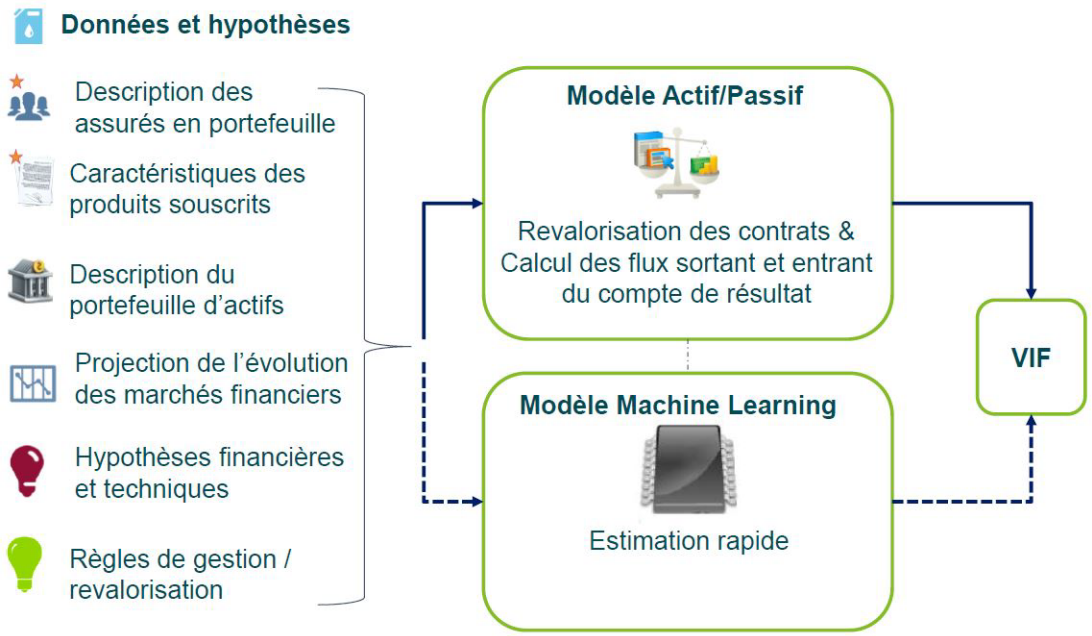



FIGURE 18 – Utilisation de la base de données pour les modèles de machine learning

Source: Mémoire IA Tarek AOUDI

### 3.2.1 Arbre de classification et de régression

Les arbres de classification et de régression (Classification And Regression Tree, CART) sont des modèles de machine learning qui se base sur des arbres afin de prédire un résultat à partir d'un jeu d'observations. Il s'agit d'arbres de classification quand la sortir à prédire est une classe appartenant aux données, et il est question d'arbres de régression quand le résultat prédit est un nombre réel. Afin de pouvoir effectuer la tâche demandée, les CART se basent sur le principe de partitionnement récursif : l'espace est sous-divisé en espaces de plus en plus petit jusqu'à devenir suffisamment simples pour être traités par des modèles plus simples.

L'arbre est alors une représentation de ce partitionnement, avec chaque noeud terminal (ou feuille) représentant une cellule du partitionnement. On a alors qu'une observation $X$ va appartenir à la feuille $x$ (et donc donner le résultat $x$) si le cheminement de l'arbre en utilisant cette observation amène à la feuille $x$.

Pour déterminer le chemin emprunté par l'observation, l'algorithme débute à la racine (le premier noeud de l'arbre) et chaque noeud sous-jacent constitue une question pour l'observation. En fonction de la réponse le cheminement continuera au noeud suivant correspondant.

Dans le cas d'un arbre de régression classique, la valeur prédite dans la feuille finale $l$ est simplement une moyenne des données de cette cellule. Soient $(x_i, y_i), i \in 1, ..., N$ des échantillons qui appartiennent à la feuille $l$, alors la valeur prédite de cette feuille est la moyenne des échantillons $\hat{y} = \sum_{i=1}^{N} \frac{y_i}{N}$

Une fois l'arbre déterminé, les modèles de chaque cellule sont fixés et facilement calculable. La difficulté est alors de trouver le meilleur arbre (et donc partitionnement) pour les données.

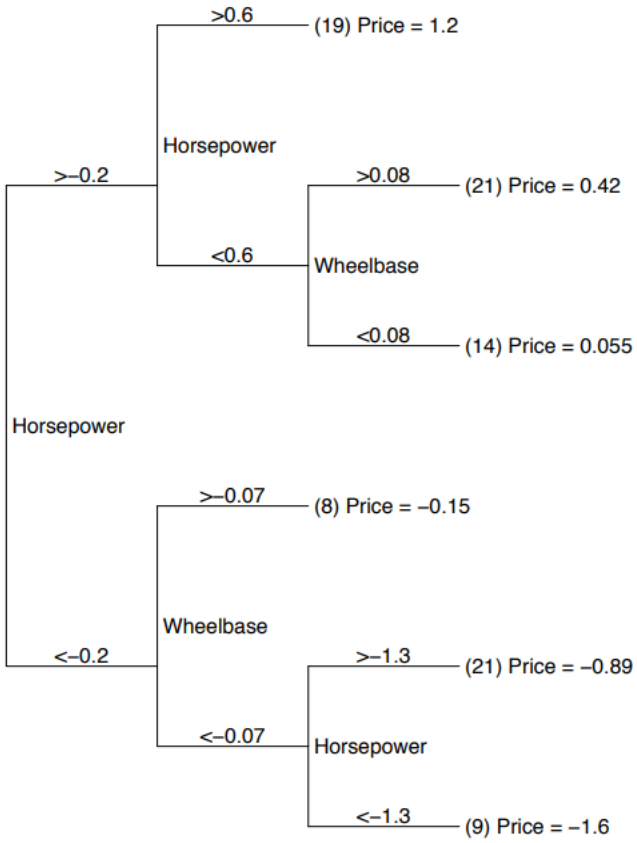



FIGURE 19 – Exemple d'un arbre de régression pour prédire le prix d'un certain type de voiture (les variables ont été centrées et réduites). Les nombres entre parenthèses indiquent le nombre d'échantillons contenue dans la cellule.

Source: Cours Statistics 36-350 : Data Mining, 2006, Carnegie Mellon University par Cosma Shalizi

Le but est d'alors de maximiser $I[C;Y]$ qui correspond à l'information qu'apporte le cluster à propos des variables explicatives $X$. Une optimisation directe n'est pas possible et la technique de la recherche gloutonne est alors utilisée. Cette technique est une optimisation étape par étape : A chaque nouvelle étape, le choix effectué est celui qui optimise le résultat de l'étape en question, sans prise en compte des potentielles répercussions sur les étapes ultérieures. La première question (et donc partition) est donc trouvée choisissant celle qui optimise l'information obtenue sur $Y$, la variable à prédire. Ces actions sont alors répétées à chaque noeud produit, cette fois en optimisant l'information sur $Y$ sachant l'information déjà connue.

Un problème surgit alors : si l'algorithme continue récursivement cette opération, il finira par atteindre un point où chaque feuille ne contient qu'un échantillon. Pour éviter cela, l'utilisation d'un critère d'arrêt permet d'arrêter de diviser le noeud si la division ne donne pas suffisamment d'informations ou que le nombre d'échantillons de la feuille descend sous un minimum fixé.

L'information qui sera souvent utilisée pour les arbres de régression est la somme des erreurs

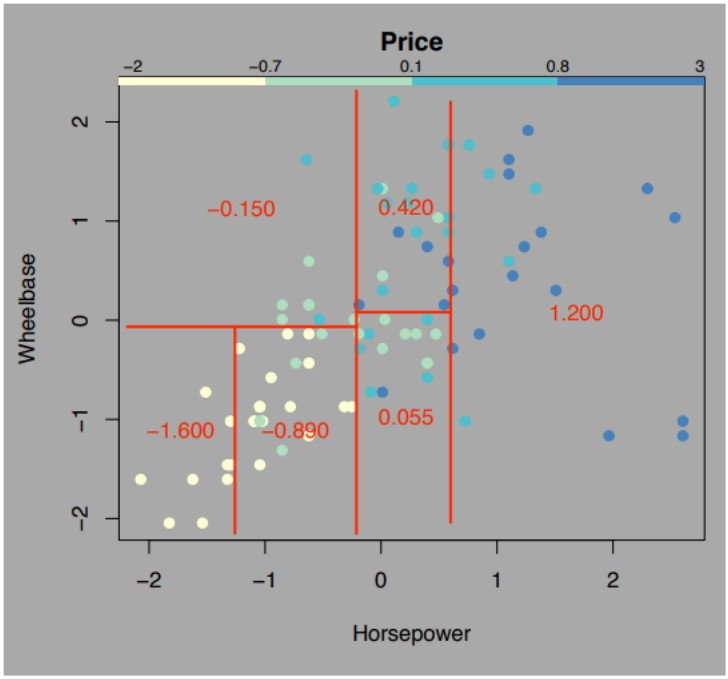



FIGURE 20 – Partitionnement des données effectué par l'arbre de régression de la figure précédente. Les lignes de séparation sont parallèles aux axes car les noeuds ne compare les données d'observation qu'à une variable à la fois.

Source: Cours Statistics 36-350 : Data Mining, 2006, Carnegie Mellon University par Cosma Shalizi

carrées (Sum of Squared Errors, SSE), qui est donnée par la formule suivante pour un arbre $T$ :

$$S = \sum_{c \in feuilles(T)} \sum_{i \in C} (y_i - m_c)^2$$

avec $m_c = \sum_{i \in C} \frac{y_i}{n_c}$ la prédiction pour la feuille $c$. Pour les arbres de classification, d'autres mesures comme l'indice d'impureté de Gini peuvent être considérées.

Une des principales limitations de cet algorithme est sa non-robustesse : une modification minime de l'ensemble d'entraînement peut mener à des modifications conséquentes des résultats prédits. Cela est très problématique étant donné qu'une des volontés de ce projet est d'être évolutif et pouvant être complété par des données passées et futures.

Pour contrer ce problème, différentes techniques ensemblistes, notamment le **bagging** et le **boosting** ont été utilisées pour améliorer les arbres de régression en de modèles bien plus puissants.

### 3.2.2 Random Forest

Les random forest (Forêts aléatoires) font partie de la famille des techniques d'apprentissage automatique et basent sur les arbres de décision et de régression et sur les principes de bagging (bootstrap aggregating). Leurs premières apparitions dans la littérature scientifique ont été par

Ho en 1995 et repris par la suite de manière plus formelle par Leo Breiman et Adele Cutler en 2001.

Le principe général est de créer de nombreux petits arbres (d'où le terme de forêts), qui auront un pouvoir prédictif moins fort qu'un grand arbre de décision seul mais, en moyennant ces petits arbres (ou en choisissant le vote majoritaire dans le cas d'une classification), il est alors possible d'obtenir un modèle très puissant, bien plus stable que les CART, moyennant la perte d'interprétabilité de ce dernier.

La logique derrière ce modèle se base sur différentes théories et informations.

En régression, l'erreur attendue d'un modèle $M$ pour un point $x$, en supposant que la fonction d'erreur $L$ correspond au SSE, peut être écrit de la manière suivante :

$$\begin{aligned} Erreur(M_L(x)) &= \mathbb{E}_{Y|X=x}[(Y - M_L(x))^2] \\ &= \mathbb{E}_{Y|X=x}[(Y - M_B(x) + M_B(x) - M_L(x))^2] \\ &= \mathbb{E}_{Y|X=x}[(Y - M_B(x))^2 + (M_B(x) - M_L(x))^2 + 2(Y - M_B(x))(M_B(x) - M_L(x))] \\ &= \mathbb{E}_{Y|X=x}[(Y - M_B(x))^2] + \mathbb{E}_{Y|X=x}[(M_B(x) - M_L(x))^2] \\ &= Erreur(M_B(x)) + (M_B(x) - M_L(x))^2 \end{aligned}$$

avec $M_B$ le modèle de Bayes qui est sans biais, et donc $\mathbb{E}_{Y|X=x}[Y - M_B(x)] = \mathbb{E}_{Y|X=x}[Y] - M_B(x) = 0$. Ici, chaque terme de l'équation obtenue correspond à une erreur spécifique. Le premier terme va correspondre à l'erreur résiduelle irréductible au point $x$ tandis que le second terme correspond à la différence entre le modèle $M_L$ et $M_B$. Plus le modèle sera performant, plus ce second terme sera petit.

Il est notamment possible de réécrire cette équation sous la forme suivante :

$$\mathbb{E}[Erreur(M_L(x))] = bruit(x) + biais(x)^2 + var(x) \tag{10}$$

qui est alors la **décomposition biais-variance** de notre erreur.

Le bruit est indépendant des données fournies au modèle et est donc irréductible. Il agit alors comme une erreur minimale pour tout modèle. Le biais mesure la différence entre le modèle bayésien et le modèle utilisé et le dernier terme correspond à la variance dans les prédictions du modèle.

La méthode du bagging va donc chercher à minimiser l'erreur pour se rapprocher au plus de l'erreur liée au bruit. Pour ce faire, l'algorithme va chercher à créer de manière aléatoire un ensemble d'arbres, et la prédiction finale de la forêt sera une moyenne des prédictions de chaque arbre. Cette moyenne va en effet conserver le biais et l'erreur liés au bruit, mais va permettre de réduire fortement la variance de nos arbres et donc de l'erreur de prévision. Les arbres sont quant à eux construits de manière aléatoire :

— n observations sont tirées aléatoirements de l'ensemble d'entraînement (tirage avec remise).

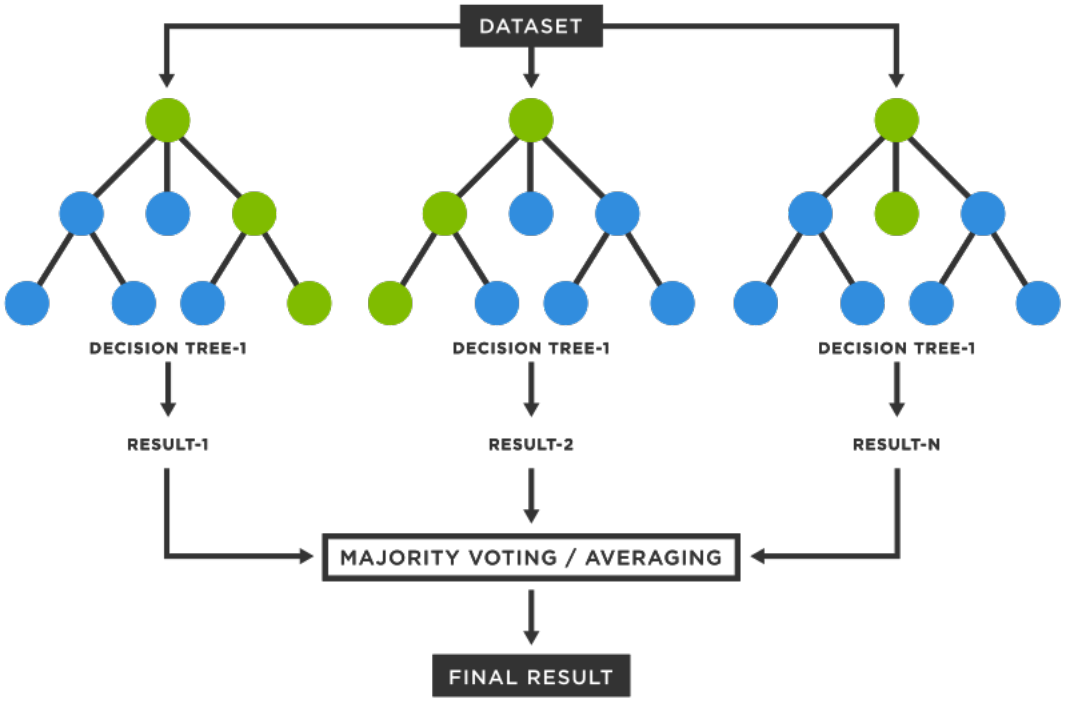



FIGURE 21 – Illustration du random forest : passage de prédictions individuelles à la prédiction finale

Source: www.tibco.com

— L'arbre considéré est alors entraîné sur ce sous ensemble

Notons que la réduction de l'erreur n'est possible que si nos arbres sont suffisamment décorrélés. En cas de corrélation forte, la variance obtenue par bagging ne sera que très peu réduite.

Le tirage d'observations aléatoires permet de réduire la corrélation entre les arbres, de même que la manière dont les variables sont choisies et coupées à chaque noeud.

Les avantages du random forest sont alors les suivants :

— Meilleure performance en générale que les CART

— Paramétrage plus facile à effectuer

— Pas de sur-apprentissage

Mais le modèle vient aussi avec son lot d'inconvénients :

— Modèle boite noire : il est difficile d'interpréter le modèle avec un nombre d'arbres aussi grand.

— L'entraînement du modèle est aussi plus lent, rendant son utilisation sur une base de données conséquente plus difficile.

### 3.2.3 eXtreme Gradient Boosting

Le eXtreme Gradient Boosting (XGBoost) est comme son nom l'indique, une méthode qui repose sur le **boosting** qui a été initiallement développé par Tianqi Chen

Les algorithmes de boosting se reposent sur la combinaison itérative de prédicteurs faibles afin d'en construire un plus précis et puissant. Dans le cas du XGBoost, la manière de créer ces prédicteurs sont semblables à celles du Gradient Boosting avec une implémentation spécifique afin de grandement accélérer la vitesse de calcul. Le Gradient Boosting construit ses prédicteurs de telle sorte qu'à chaque itération, le nouveau prédicteur créé corrige les erreurs de son prédécesseur en s'entraînant sur l'erreur résiduelle de la prédiction précédente.

Contrairement au Gradient Boosting qui fonctionne comme une descente de gradient dans l'espace de la fonction, XGBoost fonctionne comme une méthode de Newton-Raphson dans l'espace de la fonction.

Un algorithme générique pour le XGBoost serait :

Soit un ensemble d'entraînement $(x_i, y_i)$, $i = 1, ..., N$, une fonction de perte dérivable $L(y, F(x))$, un nombre de prédicteurs faibles $M$ ainsi qu'une vitesse d'apprentissage $\alpha$.

1. Initialiser le modèle avec une valeur constante :

$$\hat{f}_{(0)}(x) = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, \theta)$$

2. Pour $m = 1$ à $M$ :

   (a) Calculer les gradients et les hessiennes

$$\hat{g}_m(x_i) = \left[ \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x_i)=\hat{f}_{(m-1)}(x)}$$
$$\hat{h}_m(x_i) = \left[ \frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2} \right]_{f(x_i)=\hat{f}_{(m-1)}(x)}$$

   (b) Entraîner un arbre en utilisant l'ensemble d'entraînement $x_i, -\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}$ pour $i = 1, ..., N$ en résolvant le problème d'optimisation suivant :

$$\hat{\phi}_m = \arg\min_{\phi \in \Phi} \sum_{i=1}^{N} \frac{1}{2} \hat{h}_m(x_i) \left[ \frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)} - \phi(x_i) \right]^2$$
$$\hat{f}_m(x) = \alpha \hat{\phi}_m(x)$$

   (c) Mettre à jour le modèle :

$$\hat{f}_m(x) = \hat{f}_{(m-1)}(x) + \hat{f}_m(x)$$

3. Le modèle final est :

$$\hat{f}(x) = \hat{f}_M(x) = \sum_{m=0}^{M} \hat{f}_m(x)$$

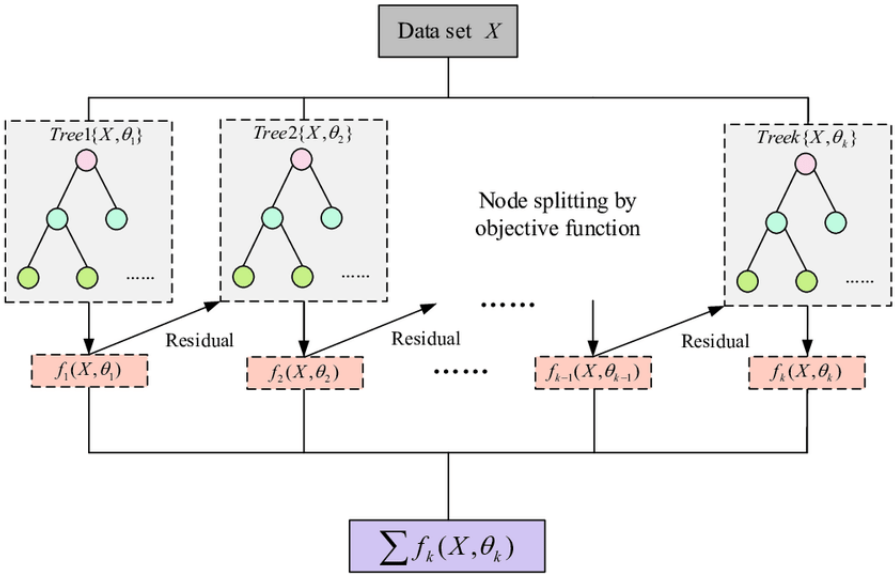



FIGURE 22 – Déroulement de l'algorithme XGBoost

Source: Degradation state recognition of piston pump based on ICEEMDAN and XGBoost, Rui Guo et al

Les principaux avantages du XGBoost sur les autres modèles sont liés à son implémentation parallélisée ainsi qu'une utilisation optimisée des ressources permettant au modèle d'être très performant avec des temps d'entraînement relativement courts. La possibilité de pénaliser le modèle via des régularisations est aussi un avantage algorithmique afin d'éviter le sur-apprentissage.

Cependant, elle vient aussi avec ses inconvénients, en particulier l'effet boîte noire qui est grandement amplifié par rapport au Random Forest. Le Random Forest peut être difficile à interpréter dû au grand nombre d'arbres, mais l'interprétation de chaque arbre restait relativement aisée. Dans le cas du XGBoost, l'interprétation des arbres de manière itérative est beaucoup plus complexe.

### 3.3 Application et tuning des hyperparamètres

L'ensemble des modèles utilisés sont issus du package *sklearn* de Python. Ce package est très complet et bien documenté et va permettre de mettre en place de manière très simple et rapide les modèles sur lesquels les données seront appliquées. De plus, il comprend de nombreuses fonctions permettant d'optimiser et tester nos modèles qui seront utilisés dans le cadre de ce mémoire.

Les modèles utilisés sont très puissants et disposent de nombreux paramètres qui permettent de modifier leur comportement afin qu'ils puissent mieux s'adapter aux données qui leur sont présentées. Ces paramètres sont appelés **hyperparamètres** et jouent un rôle important dans la complexité des différents modèles. Il est alors nécessaire de porter une attention particulière à ces derniers afin de les choisir de sorte à ce qu'ils soient les plus performants possibles.

### 3.3.1 Le Grid Search

Une des techniques d'optimisation qui sera utilisée est le grid-search (recherche par grille en français). Comme son nom l'indique, cette technique se base sur l'idée de quadriller l'espace des hyperparamètres possibles.

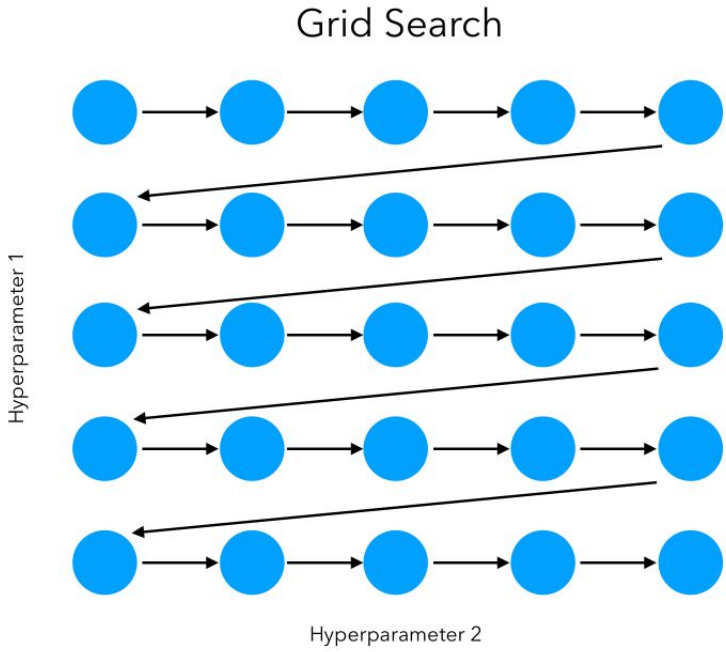



FIGURE 23 – Illustration du grid-search sur deux hyperparamètres

Source: A Guide to Hyperparameter Optimization (HPO), Maël Fabien Github

Supposons que le modèle à tester dispose de deux hyperparamètres à optimiser $HP_1$ et $HP_2$. Pour chacun de ces paramètres, une plage des valeurs du paramètre possible doit être fixée. De manière général, une borne inférieure, une borne supérieure ainsi qu'une taille de plage sont définies, et les valeurs de la plage sont alors réparties de manière uniforme sur cette plage.

$$Plage_{HP_1} = \left\{ i \times \frac{(V_{max} - V_{min})}{N} + V_{min}, i \in [[0, N]] \right\}$$

Avec $V_{min}$ la borne inférieur, $V_{max}$ la borne supérieur et $N+1$ le nombre de valeurs de la plage.

Chaque jeu (ici pair) de paramètres est alors testé l'un après l'autre (voir fig 23) et la performance du modèle ainsi paramétré mesurée pour finalement choisir le jeu de paramètres ayant la meilleure performance. Souvent, cette mesure de performance est effectuée avec un **validation croisée**, processus qui sera expliqué en section 3.3.3, et une fonction de perte ou d'erreur.

L'avantage de cette méthode est qu'elle permet de tester l'intégralité des paramètres possibles (dans la limite des plages définies) en simultanée. Il est alors possible de se rapprocher au maximum du jeu de paramètres optimal pour le modèle contrairement à une optimisation qui serait faite paramètre par paramètre où le jeu de paramètres obtenu serait une fonction de nombreuses variables extérieurs.

Cependant, le plus grand inconvénient de cette méthode est la puissance de calcul nécessaire pour optimiser le modèle dans un temps acceptable. En effet, le fait de tester l'ensemble des paramètres en même temps contrairement à l'un après l'autre mène à un temps de calcul qui n'est plus linéaire en fonction du nombre de paramètres (et de la plage testée) mais exponentielle en le nombre de paramètres.

Si :

— $HP_1, HP_2, ..., HP_M$ correspondent aux $M$ hyperparamètres,

— $N_1, N_2, ..., N_M$ le nombre d'éléments dans leurs plages respectives.

— $\Phi(HP_1, HP_2, ...HP_M, N_1, N_2, ..., N_M)$ le temps pour entrainer et tester le modèle en fonction de ce jeu de paramètres.

— $\mathscr{P}$ un ensemble de dimension $\prod_{i=1}^{M} N_i$ contenant l'ensemble des jeux d'hyperparamètres possible étant donné nos bornes et nombres de valeurs dans les plages.

Alors le temps mis par le grid search pour trouver le jeu de paramètres optimal peut être approché par (ces équations sont à titres illustratifs pour avoir une idée des ordres de grandeurs en jeu) :

$$Temps = \prod_{(HP_1, HP_2, ...HP_M) \in \mathscr{P}} \Phi(HP_1, HP_2, ...HP_M, N_1, N_2, ..., N_M)$$

et donc en encadrant $\Phi$ par des constantes $c$ et $C$, on obtiendrait :

$$C^{\prod_{i=1}^{M} N_i} \geq Temps \geq c^{\prod_{i=1}^{M} N_i}$$

Il est alors nécessaire de choisir avec attention les hyperparamètres à optimiser ainsi que les plages de valeurs utilisées.

### 3.3.2 Tuning itératif et algorithmes gloutons

Le principal inconvénient du grid-search est le fait que sa complexité est exponentielle en le nombre de variables/combinaisons à tester. De ce fait, il est difficile d'avoir une estimation précise des paramètres optimaux et il est possible de les manquer à cause d'une plage de valeurs pas assez grande ou trop grossière. La méthode de tuning "itératif" va essayer d'approcher le problème par un nouvel oeil, en s'inspirant de la logique des algorithmes gloutons.

Les algorithmes gloutons sont des types d'algorithmes qui vont chercher à résoudre un problème en effectuant des optimisations locales. Le sens de glouton provient alors du fait que l'algorithme ne va pas essayer d'optimiser le problème de manière globale mais choisir le chemin qui, à chaque étape, lui donne la plus grande satisfaction (c'est à dire le meilleur score). Un exemple simple à cet algorithme est la stratégie gloutonne pour le problème du voyageur de commerce, où un commerçant doit visiter l'intégralité d'un ensemble de villes mais en minimisant la distance totale parcourue et en ne passant par une ville une seule et unique fois. Le choix de l'algorithme glouton sera alors, pour un point de départ donné, de se rendre à la ville directement la plus proche et ainsi de suite, jusqu'à avoir visité toutes les villes.

Une manière de voir les choses est que l'algorithme va essayer de suivre le chemin où la pente (positive) du score est la plus grande, et donc le chemin où l'augmentation du score immédiat est le plus grand.

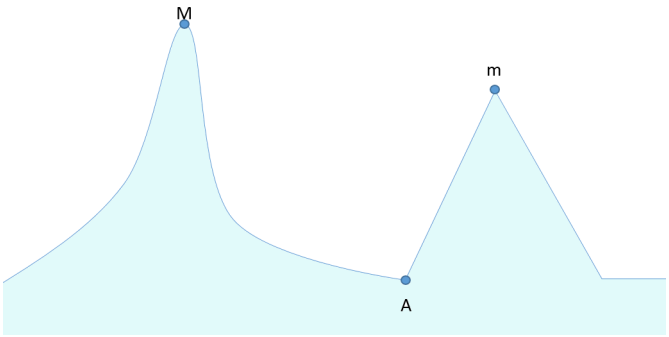



FIGURE 24 – Illustration d'une manière dont l'algorithme glouton pourrait se tromper et ne pas atteindre l'optimum global

Le graphique 24 permet de comprendre les erreurs qui peuvent être commises par ces types d'algorithmes. Bien que l'optimum soit en $M$, la pente menant à l'optimum local $m$ est plus grande que celle qui mène en $M$ (du moins localement), et l'algorithme va donc choisir ce chemin plutôt que l'autre.

L'un des inconvénients de cette méthode est que, dans le cadre des modèles utilisés, le nombre d'hyperparamètres est assez important. Cela implique alors que la dimension globale du problème augmente avec chaque paramètre. Cette dimension plus grande mène alors à un plus grand risque d'existence d'optimum locaux et donc d'obtenir un jeu de paramètres final sous optimal.

Par contre, contrairement au grid-search, la recherche des paramètres optimaux n'est plus en complexité exponentielle par rapport au nombre de paramètres ou de leurs plages de valeur, mais va plutôt se rapprocher d'une complexité linéaire.

### 3.3.3 La Validation Croisée

La validation croisée (cross-validation, CV) est une méthode de validation pour estimer la performance d'un modèle ou d'une analyse lorsqu'on la généralise à des données nouvelles. Cette méthode est issue d'un besoin de calculer la performance du modèle sans que cette performance ne soit liée à la base d'apprentissage et de validation. En effet, si la validation du modèle ne s'effectue que sur un ensemble d'entraînement et de test fixes, alors il y a un risque que le modèle soit très performant sur cette base mais devienne médiocre sur une base nouvelle à cause d'un sur-apprentissage et d'un biais de sélection (par exemple, la base d'apprentissage ne comporte pas de sensibilité sur les taux, et donc le modèle apprendra mal à réagir face à des changements de taux soudains et importants). Pour limiter ces effets, l'idée est de ne pas calculer la performance d'un modèle via un unique jeu de données entraînement/test, mais de prendre par exemple la moyenne sur plusieurs jeux de données. Il existe plusieurs méthodes de créer ces différents jeux de données, avec des méthodes exhaustives (l'intégralité des différentes combinaisons sont testées) ou non exhaustives. Cependant, la plus courante est la **k-fold**

**cross validation**.

Cette méthode consiste à diviser la base de données totale une partition de $k$ sous-ensembles de tailles égales. Chacun de ces sous-ensembles va servir de base de test, tandis que les $k-1$ autres serviront de base d'entraînement. Ce processus est alors répété au total $k$ fois et les résultats sont alors moyennés pour obtenir une unique estimation de la performance du modèle. Cette méthode a l'avantage d'utiliser l'intégralité des données dans les bases d'entraînement et test, ainsi que de n'utiliser les données une seule et unique fois dans la base de test. Le paramètre $k$ est un paramètre non fixé, généralement entre 5 et 10.

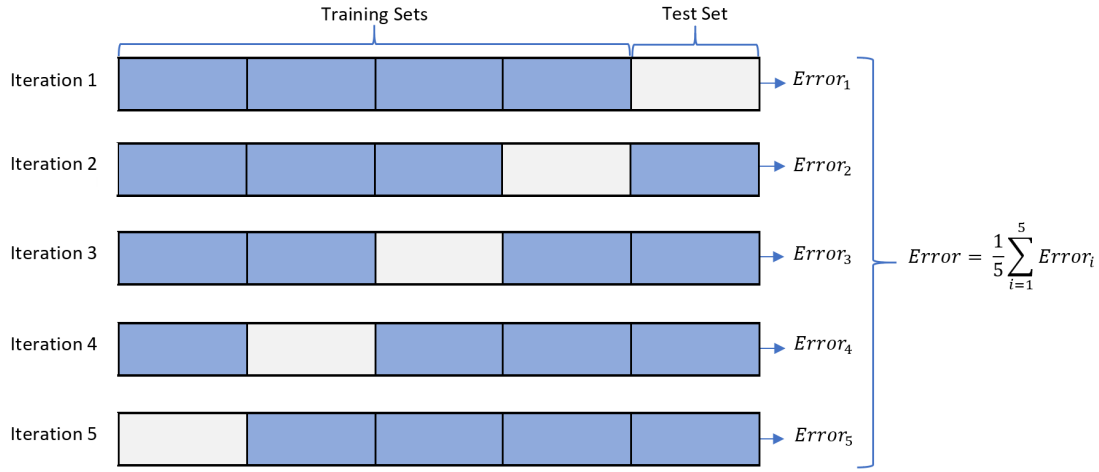



FIGURE 25 – Illustration d'une 5-fold validation

Source: towardsdatascience.com

Ces méthodes permettent alors d'avoir un estimateur presque sans biais (le biais est du au fait que la base d'entraînement est plus petite que la base réelle) pour la performance attendue du modèle sur un échantillon indépendant provenant de la même population, et le biais restant provient du fait que la base d'entraînement a une taille différente de la base finale.

Cependant, cette technique a un coût : le modèle est entraîné et testé plusieurs fois, et donc de manière évidente, le coût computationnel est multiplié par autant que le nombre de blocs utilisés pour la validation croisée. Pour peu que cette méthode soit utilisée à répétition, le coût pourrait alors devenir rapidement très important.

### 3.3.4 Random Forest

#### Les hyperparamètres du Random Forest

Les hyperparamètres du Random Forest sont nombreux et l'optimisation complète serait beaucoup trop demandante en puissance de calcul pour un résultat sensiblement identique. Les temps d'entraînement pour un jeu d'hyperparamètres limité pouvait déjà se compter en dizaines d'heures. Rappelons que la complexité du grid-search est exponentielle par rapport aux nombre d'hyperparamètres à optimiser. Nous allons alors optimiser tout ou partie de la liste des hyperparamètres suivant selon le type d'optimisation :

— *max_depth* : Profondeur maximale pour chaque arbre. Cette profondeur peut ne pas être atteinte si des contraintes provenant des autres paramètres l'en empêchent. Dans le cas contraire, un trop grande profondeur peut mener à du sur-apprentissage.

— *min_samples_split* : Nombre d'échantillons minimal dans un noeud pour pouvoir être de nouveau divisé. Une trop faible valeur peut aussi mener à du sur-apprentissage.

— *max_leaf_nodes* : Nombre maximal de feuilles qu'un arbre peut avoir. Cela permet d'éviter que bien qu'ayant un *min_samples_split* petit, d'avoir l'ensemble des feuilles avec très peu d'échantillons et donc un grand nombre de feuilles.

— *min_samples_leaf* : Nombre minimal d'échantillons dans une feuille qui permet notamment d'augmenter la généralisation du modèle en ne créant pas de chemin pour les résultats rares.

— *n_estimators* : Nombre d'arbres dans notre modèle (weak learner). Généralement, un plus grand nombre d'arbres mène à un modèle plus performant jusqu'à que l'erreur se stabilise et que les gains ne soient alors plus perceptibles. Le nombre d'estimateurs permet aussi d'augmenter la robustesse du modèle. Le temps de calcul augmente linéairement par rapport à ce paramètre.

#### Optimisation des paramètres par grid search

Afin d'effectuer une optimisation par grid-search, les plages et paramètres choisis sont les suivants :

— *max_depth* avec les valeurs suivantes : $\{10, 50, 100, 150, 300\}$

— *min_samples_split* avec les valeurs suivantes : $\{2, 5, 10, 50\}$

— *n_estimators* avec des valeurs allant de 100 à 1000 par incrément de 100.

Ces paramètres ont été choisi car étant facilement interprétables et ayant un fort impact sur la performance du modèle.

La fonction de score choisie est le negative MAE (Mean Absolute Error) :

$$negMAE = -\frac{1}{N}\sum_{i=1}^{N}|Y_i - P_i|$$

avec $N$ la taille de l'échantillon test, $Y_i$ la valeur réelle et $P_i$ la valeur prédite.

Le MAE a été choisi car étant facilement interprétable contrairement à d'autres mesures comme le RMSE (Root Mean Squared Error) qui pénalise fortement les grands écarts, ou le MAPE (Mean Absolute Percentage Error) qui peut être trompeur lorsque les valeurs absolues sont très faibles.

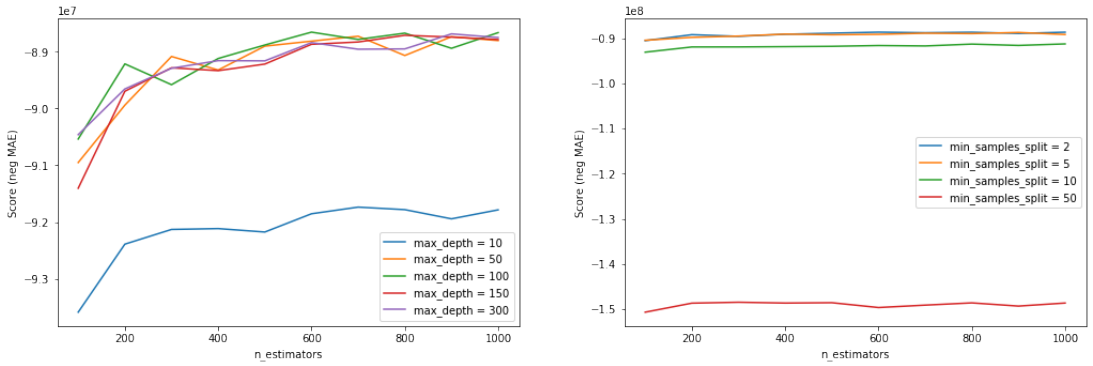
La partie "négative" de ce MAE sert à le faire passer d'une fonction d'erreur (MAE) à une fonction de scoring (neg MAE). Les algorithmes utilisant les fonctions d'erreur auront pour but de minimiser ces derniers tandis que les algorithmes utilisant des fonctions de scoring voudront les maximiser.

Clairement, les plages choisies ne couvrent pas suffisamment l'espace pour déterminer de manière précise les optimums, et les bornes des plages peuvent être limitées (notamment pour le nombre d'estimateurs) mais ce choix a été fait afin de préserver un processus qui pourra être répliqué de manière convenable. En effet, avec les plages actuelles, cela résulte à un total de 200 combinaisons différentes, et étant donné que pour la validation croisée, un 3-Fold a été choisi (80% de la base en tant que base d'entraînement et 20% en tant que base test), le nombre de fit (entraînement du modèle) total est de 600. Le temps d'un entraînement varie selon les paramètres entre 17 secondes et 2 minutes, ce qui donne finalement un temps d'optimisation compris entre **170 minutes** et **20 heures**.

Afin de pouvoir mieux comprendre l'impact des paramètres sur le score obtenu, deux types de graphiques ont été produits :

— Des graphiques en 2D multilignes représentant l'évolution du score en fonction de l'hyperparamètre *n_estimators*. Chaque ligne représente une valeur du 2nd paramètre varié. Les autres paramètres sont fixés à leur valeurs dans le jeu de paramètre optimal obtenu par grid search.

— Des graphiques en 3D permettant de faire varier à la fois le paramètre *n_estimators* et le second paramètre. L'axe Z (hauteur) représentera alors le score pour ce jeu de paramètre.




(a) Profondeur maximale (b) Nombre d'échantillon minimal pour split

FIGURE 26 – Évolutions du neg MAE par rapport au nombre d'estimateurs et des autres paramètres

Sur le graphique (a), on voit clairement l'amélioration du score en fonction du nombre d'estimateurs. L'augmentation du nombre d'estimateurs est généralement corrélée avec une meilleure

prédiction, la variance étant réduite avec un grand nombre d'estimateur, l'erreur va alors tendre vers le biais résiduel. Cependant, le score finit par plafonner à partir de 800 estimateurs. Concernant la profondeur maximale à choisir, peu d'informations peuvent être tirées de ce graphique. Une profondeur faible de 10 n'est pas suffisante pour obtenir des bonnes prédictions mais les résultats pour une profondeur supérieure à 50 sont relativement équivalents.

Dans le (b) l'information est encore plus difficile à interpréter. En effet, à l'exception du *min_samples_split* = 50 qui affiche un score médiocre, les autres semblent mener à des résultats très similaires. De plus, la variable ne semble être que très peu affectée par le nombre d'estimateurs du modèle.

Afin de pouvoir avoir une meilleure idée de l'impact des paramètres sur le score, il est nécessaire de visualiser l'information dans sa globalité, c'est à dire de ne plus fixer le paramètre non affiché afin de pouvoir apprécier l'impact croisé des variables entre elles.

Comme il est difficilement faisable de représenter sur un seul graphique des variations sur 3 axes ainsi qu'un résultat, une variation sur 2 axes a été choisie et le paramètre ayant la plus petite plage de valeurs sera varié en créant plusieurs graphiques.

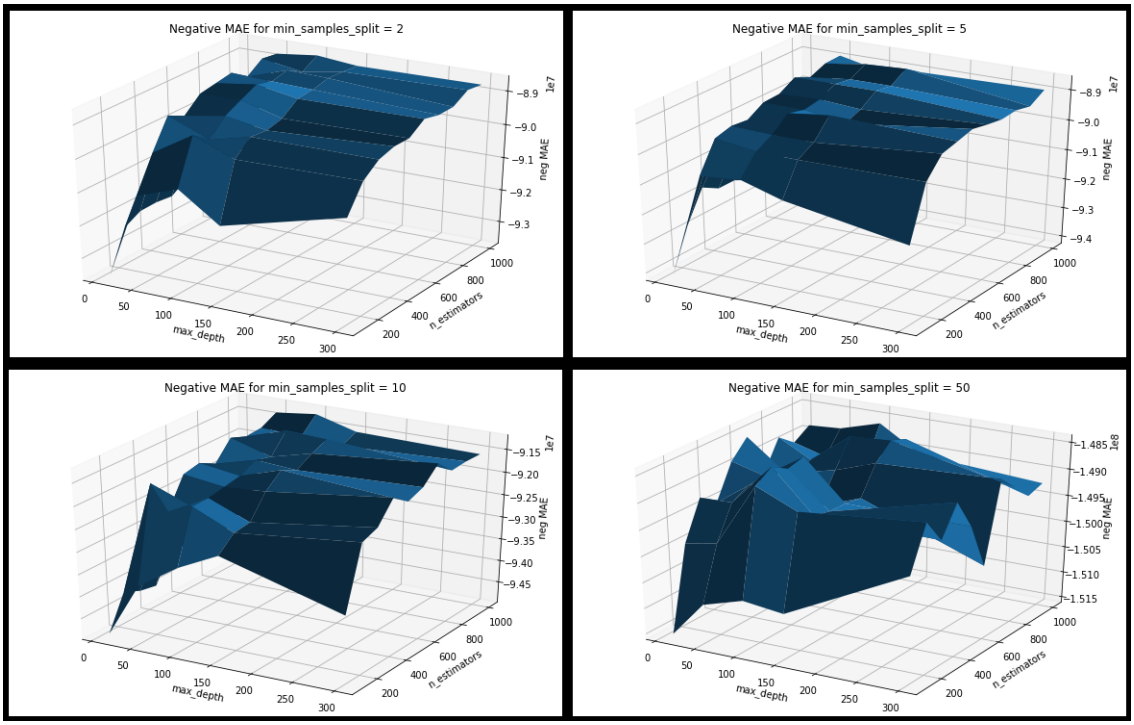



FIGURE 27 – Visualisation en 3D du score en fonction de l'ensemble des paramètres

Lors des précédents graphiques, la profondeur maximale considérée était de 100 et c'est bien pour cette valeur que le score est le plus stable en fonction du nombre d'estimateurs et ce, peu importe la valeur du *min_samples_split*. Il est maintenant cependant possible de remarquer que pour des profondeurs trop faibles ou au contraire, trop grandes, la variabilité du score augmente fortement. Cela est d'autant plus vrai pour *min_samples_split* = 50.

Les meilleures scores restent cependant assez proches. Bien qu'un jeu de paramètres optimal a été choisi, il est possible qu'il ne soit pas celui qui donnera le meilleur résultat pour le modèle

à cause à la fois du manque de granularité sur les plages de valeur des paramètres ainsi que de la faible portion de la base de données finale utilisée pour l'optimisation.

Le jeu de paramètres optimal par grid search pour le Random Forest est alors le suivant :

— *max_depth* : 100

— *min_samples_split* : 2

— *n_estimators* : 600

**Optimisation des paramètres de manière itérative**

La deuxième méthode appliquée pour l'optimisation des paramètres est la méthode itérative se basant sur le principe des algorithmes gloutons. Cette méthode étant relativement plus rapide et efficace que le grid-search, il est possible de tester plus d'hyperparamètres, ou alors de considérer des plages de valeur plus grandes ou plus précises. Il faut cependant raisonnable, la complexité linéaire énoncée plus haut étant une estimation et ne prend pas compte des constantes qui peuvent être potentiellement plus importantes dans ce cadre-là. La proportion de la base finale utilisée est de **10%** pour le Random Forest.

Dans le cadre du tuning itératif pour le Random Forest, les paramètres et les plages utilisés sont les suivants :

— *max_depth* avec les valeurs suivantes : $\{5 \times i, i \in [\![1,9]\!]\}$

— *min_samples_split* avec les valeurs suivantes : $\{[\![2,10]\!], 100, 500, 1000, 10000\}$

— *max_leaf_nodes* avec les valeurs suivantes : $\{5 \times i, i \in [\![1,9]\!]\}$

— *min_samples_leaf* avec les valeurs suivantes : $\{[\![1,5]\!], i \times 10, i \times 500, i \in [\![1,9]\!]\}$

— *n_estimators* avec des valeurs allant de 100 à 1425 par incrément de 75.

Les paramètres seront optimisés selon l'ordre ci-dessus. Le choix de l'ordre de l'optimisation des variables a une incidence sur le jeu de paramètres final obtenu. L'idée est alors d'optimiser les variables les plus impactantes en premier car ce sont celles qui feront approcher l'algorithme de l'optimum le plus rapidement. Une exception est faite pour le paramètre *n_estimators* qui est optimisé en dernier car ayant un effet très important sur la vitesse d'optimisation du modèle.

Les paramètres étant optimisés un à un, il est possible d'obtenir des graphiques résumant l'évolution de l'erreur en fonction des valeurs des paramètres. Les paramètres sont mis à jour au fur et à mesure, les paramètres trouvés sont alors utilisés comme nouveaux paramètres de base lors de l'optimisation du paramètre suivant.

Un constat marquant de ce graphique est que la différence des erreurs entre l'ensemble d'entraînement et celui de test est assez faible, et à la même allure. Une explication pour ce comportement est la forte similarité de la base d'entraînement et de test. Bien que seule une sous base de données ait été utilisée, elle a été construite de manière aléatoire. Des scénarios de la même sensibilité pour le même choc ont alors pu se retrouver à la fois dans l'ensemble d'entraînement et de test. Deux scénarios sont généralement assez semblables, les chocs apportant les plus gros effets "directs" sur la VIF.

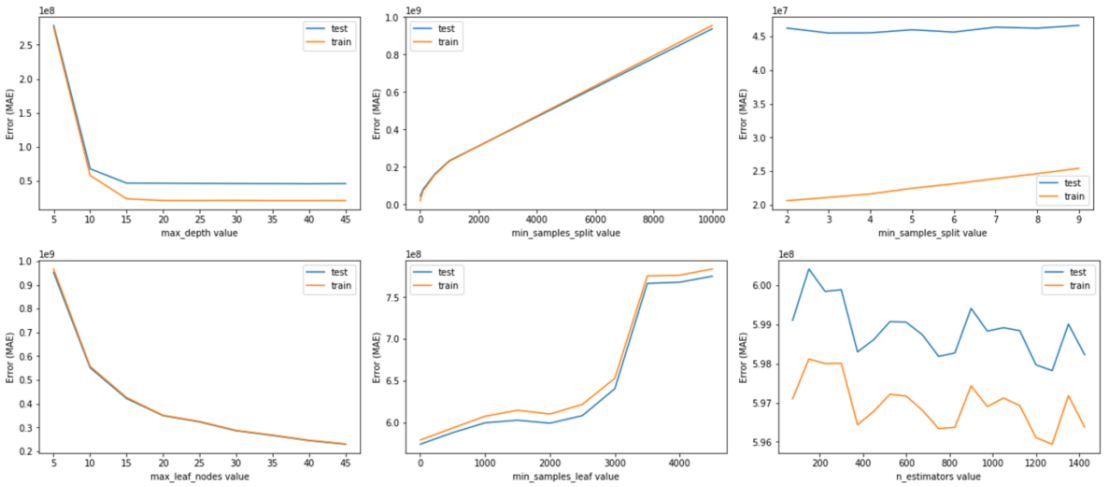



FIGURE 28 – Évolution de l'erreur en fonction des paramètres lors de l'optimisation itérative

Pour les paramètres optimisés, le *max_leaf_nodes* et le *n_estimators* sont les seuls qui ont potentiellement une valeur optimale qui n'a pas été testée. En effet, pour ces deux paramètres, la courbe est décroissante en fonction du paramètre et est coupée avant de se stabiliser comme c'est le cas pour *max_depth*. Des paramètres plus performants peuvent alors être possiblement trouvés en augmentant la plage de ces deux paramètres. A l'opposé, les trois autres hyperparamètres ont une erreur croissante en fonction de la valeur du paramètre. Les optimums pour la base d'entraînement utilisée sont alors bien définis et connus.

La réduction de l'erreur en fonction de l'optimisation des paramètres est aussi assez minime. Cela peut s'expliquer par le fait que les effets croisés entre les hyperparamètres peuvent être assez importants : l'optimisation des premiers paramètres va donc beaucoup influer sur l'erreur du modèle, sa capacité à apprendre de la base et donc à prédire correctement le modèle. Les paramètres suivants auront alors un impact moindre ou pire, auront tendance à tendre vers les valeurs non optimales.

Les paramètres obtenus via cette méthode sont alors les suivants :

— *max_depth* : 40

— *min_samples_split* : 2

— *max_leaf_nodes* : 9

— *min_samples_leaf* : 1

— *n_estimators* : 1275

En comparant avec les paramètres obtenus avec le grid-search, le seul paramètre ayant la même valeur est le *min_samples_split* qui est à sa valeur minimale.

La profondeur deux fois et demi plus grande pour le grid-search, mais le nombre d'estimateurs plus de deux fois moins élevé. En ce qui concerne les paramètres non modifiés pour le grid search, leurs valeurs de base sont les suivantes :

— *max_leaf_nodes* : None (illimité)

— *min_samples_leaf* : 1

Le second paramètre est donc identique mais le premier est grandement différent. Ce paramètre est d'ailleurs assez performant pour réduire l'erreur en se basant sur la figure 28 et donc il n'est pas incohérent de penser que le paramètre issue du grid-search soit plus performant que celui du tuning itératif.

### 3.3.5 eXtreme Gradient Boosting

#### Les hyperparamètres du XGBoost

Le XGBoost a été plus facile à optimiser sur nos machines, permettant l'utilisation d'un plus grand nombre d'hyperparamètres et de plus grandes plages de valeurs. Le choix des hyperparamètres a été fait selon la même philosophie que pour le Random Forest : interprétabilité des hyperparamètres et impact sur le résultat final.

Les hyperparamètres qui seront retenus sont les suivants :

— *max_depth* : Profondeur maximale pour chaque arbre. Cette profondeur peut ne pas être atteinte si des contraintes provenant des autres paramètres l'en empêchent. Dans le cas contraire, un trop grande profondeur peut mener à du sur-apprentissage.
— *learning_rate* : Vitesse d'apprentissage qui permet de réduire les poids de nouvelles features après chaque pas de boosting. Cela permet à l'algorithme d'être plus conservatif et d'éviter l'overfitting (bien que ralentissant la phase d'entraînement).
— *min_child_weight* : Somme des poids (hessienne) minimale dans une feuille pour la séparer. Plus ce paramètre est grand, plus l'algorithme est conservatif.
— *gamma* : Réduction de perte minimale pour séparer une feuille dans l'arbre en deux. Plus gamma est grand, plus l'algorithme est conservatif.
— *subsample* : Proportion des données qui vont être échantillonné aléatoirement avant de créer les arbres pour éviter l'overfitting. Cet échantillonnage est effectué pour chaque arbre.
— *colsample_bytree* : Identique à subsample mais pour les colonnes (features).
— *reg_alpha* : Régularisation L1 sur les poids. Plus ce paramètre est grand, plus l'algorithme est conservatif.
— *n_estimators* : Nombre d'arbres dans notre modèle (weak learner). Généralement, un plus grand nombre d'arbres mène à un modèle plus performant jusqu'à que l'erreur se stabilise et que les gains ne soient alors plus perceptibles. Le nombre d'estimateurs permet aussi d'augmenter la robustesse du modèle. Le temps de calcul augmente linéairement par rapport à ce paramètre.

#### Optimisation des paramètres par grid search

Pour l'optimisation par grid-search du XGBoost, les paramètres et plages qui seront utilisés sont les suivants :

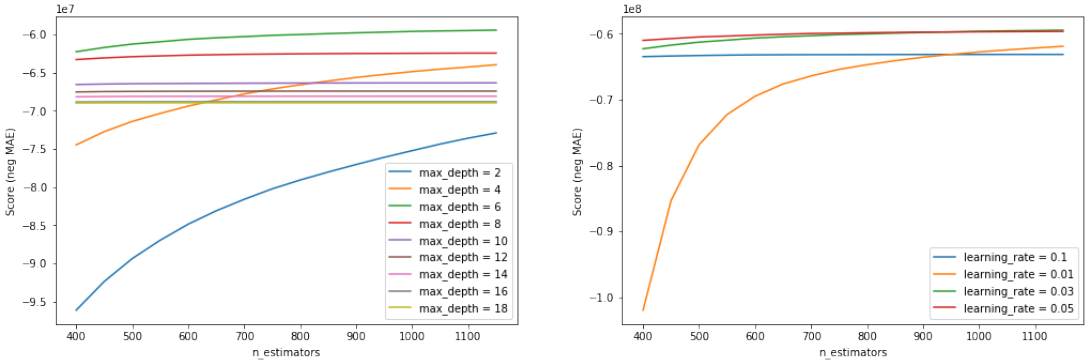
— *max_depth* avec les valeurs suivantes : $\{2 \times i, i \in [[1,10]]\}$.

— *n_estimators* avec des valeurs allant de 400 à 1200 par pas de 50.

— *learning_rate* avec des valeurs dans $\{0.1, 0.05, 0.03, 0.01\}$

Pour le XGBoost, le nombre d'estimateurs est optimisé après la vitesse d'apprentissage car ce dernier a un impact très important sur la vitesse d'entraînement et d'exécution du modèle.

Comme pour le Random Forest, la fonction de score choisie est le neg MAE et des choix ont été faits sur les paramètres à optimiser et les plages afin de ne pas surcharger l'optimisation du modèle.




(a) Profondeur maximale

(b) Vitesse d'apprentissage

FIGURE 29 – Évolutions du neg MAE par rapport au nombre d'estimateurs et des autres paramètres

Dans le graphique (a), la profondeur maximale idéale serait 6, étant celui donnant le meilleur résultat peu importe le nombre d'estimateurs. Il est clair que le modèle est plus sensible aux valeurs "faibles" de profondeurs maximales étant les seules marquant une progression du score avec l'augmentation du nombre d'estimateurs. Pour la plupart des autres valeurs de *max_depth*, le score ne varie quasiment pas et ce peu importe le nombre d'estimateurs.

Un phénomène semblable est observable dans le graphique (b) pour les différentes valeurs de taux d'apprentissage, avec néanmoins un léger effet du nombre d'estimateurs. L'unique exception est pour *learning_rate* = 0.01 où on a une forte progression du score avec l'augmentation du nombre d'estimateurs.

En ce qui concerne le taux d'apprentissage, les résultats pour une valeur de 0.05 et 0.03 sont très proches pour des grands nombres d'estimateurs. La courbe pour 0.1 a elle tendance à s'écraser et à stagner, signe que l'effet de généralisation est trop grand et que le modèle a tendance à sous-apprendre. Enfin, le modèle ayant une vitesse d'apprentissage de 0.01 n'est pas aussi bon que les deux premiers. Une raison pour cela est que, étant donné la vitesse d'apprentissage plus faible, le nombre d'estimateurs est insuffisant pour profiter pleinement de cette vitesse d'apprentissage plus lente mais il est possible de deviner sa croissance potentielle en augmentant le nombre d'estimateurs.

Pour *eta* = 0.1 (*eta* étant un nom couramment donné au *learning_rate*), l'effet sous-apprentissage "plafond" est facilement observable. En effet, peu importe le nombre d'estimateurs, le score

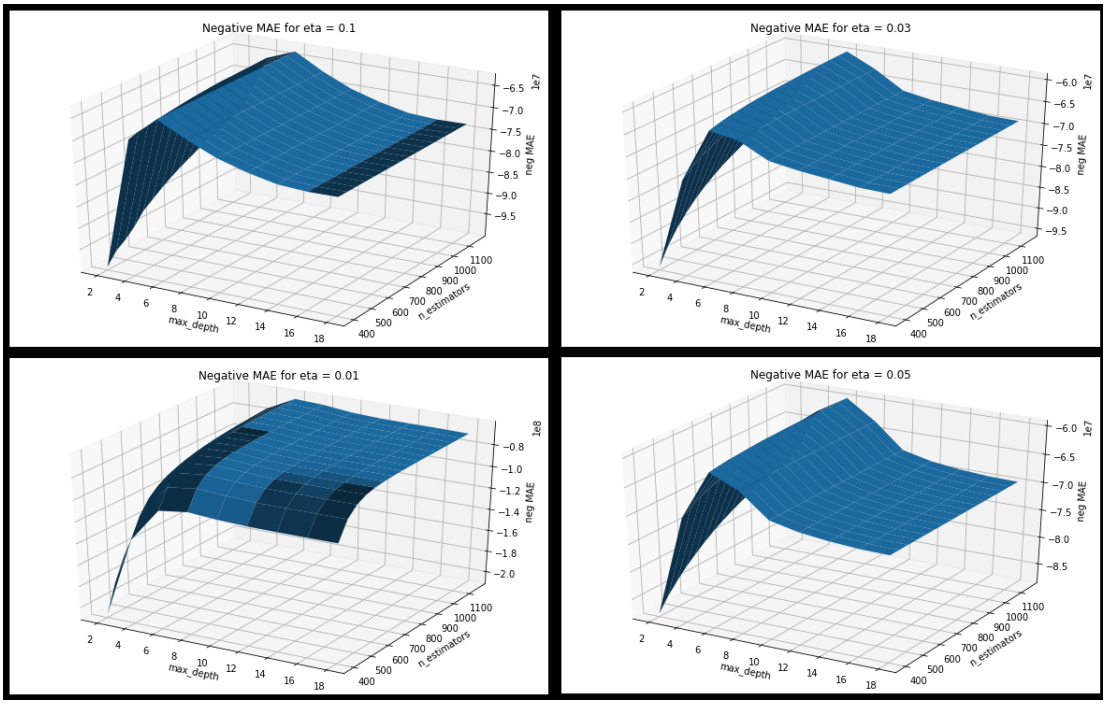



FIGURE 30 – Visualisation en 3D du score en fonction de l'ensemble des paramètres

reste le même et ne dépend que de la profondeur maximale des arbres. Le nombre d'estimateurs le plus bas dépasse donc déjà le nombre d'arbres maximal à partir duquel le modèle ne s'améliore plus pour la vitesse d'apprentissage donnée.

L'effet du nombre d'estimateurs sur *eta* = 0.03 et *eta* = 0.05 est aussi relativement faible, concordant alors avec les observations des précédents graphiques. De manière générale, la vitesse d'apprentissage n'a que peu d'effet sur le score. Cela peut être notamment dû à une trop faible quantité de données, les paramètres ne pouvant alors pas être exploités totalement.

Pour les données utilisées, on a cependant une démarcation pour *max_depth* = 4,6 qui obtiennent de meilleurs résultats que les autres profondeurs maximales. Le jeu de paramètres obtenu à l'issue de ce grid-search est alors :

— *max_depth* : 6

— *n_estimators* : 1150

— *learning_rate* : 0.03

**Optimisation des paramètres de manière itérative**

De la même manière que pour le Random Forest, une optimisation itérative sera effectuée afin de pouvoir optimiser un plus grand nombre de paramètres ainsi que d'utiliser des plages de valeur plus fine. Contrairement au Random Forest, l'optimisation itérative du XGBoost sera mêlée avec du grid-search. L'objectif est alors d'optimiser un plus petit nombre de paramètres de manière itérative, et parfois même d'optimiser des paramètres plusieurs fois (notamment le *n_estimators*) afin de prendre en compte au fur et à mesure des nouveaux paramètres.

L'optimisation du XGBoost étant plus rapide que celle du Random Forest, 20% de la base de données initiale sera utilisée pour cette partie.

Les paramètres optimisés de cette manière et leurs plages sont les suivants :

— *n_estimators* avec des valeurs allant de 1 à 600.

— *max_depth* avec des valeurs dans $\{3+2\times i, i\in[[0,5]]\}$

— *min_child_weight* avec des valeurs dans $[[1,5]]$

— *gamma* avec des valeurs allant de 0 à 4 avec un pas de 0.4

— *subsample* avec des valeurs allant de 0.1 à 0.9 avec un pas de 0.1

— *colsample_bytree* avec des valeurs allant de 0.1 à 0.9 avec un pas de 0.1

— *reg_alpha* avec des valeurs dans $\{1e-5, 1e-2, 0.1, 1, 50, 100, 1000\}$

— *learning_rate* avec des valeurs dans $\{0.01, 0.03, 0.05, 0.1, 0.3\}$

Les paramètres suivants seront optimisés ensembles par un grid-search :

— *max_depth* et *min_child_weight*

— *subsample* et *colsample_bytree*

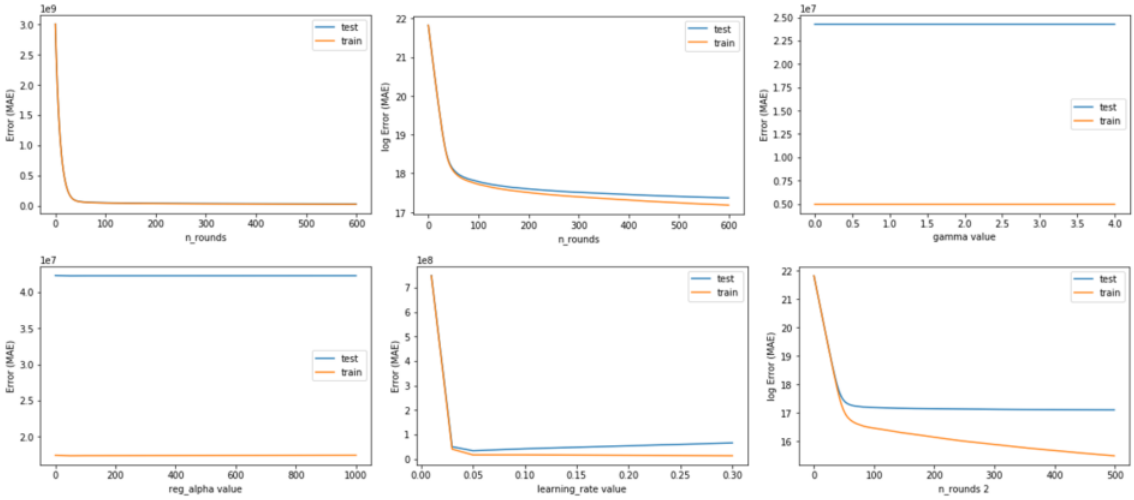



FIGURE 31 – Optimisation des variables du XGB de manière itérative (ordre de gauche à droite, de haut en bas)

L'analyse variable par variable permet rapidement de repérer les paramètres n'ayant pas un impact important sur l'erreur. Les paramètres *gamma* et *alpha* sont des paramètres de régularisation qui ne sont pas impactant dans le contexte des autres paramètres et de la base de données utilisée.

L'erreur liée au nombre d'estimateurs *n_rounds* semble se stabiliser très rapidement que ce soit pour la première optimisation ou pour la seconde. En passant par le logarithme il est possible de mieux apprécier la différence entre l'ensemble d'entraînement et de test. Lors de la première optimisation, la baisse de l'erreur pour l'ensemble test est faible mais encore présente, le modèle aurait alors pu être mieux optimisé avec une plage de valeur différente. La seconde

optimisation ne lui permet pas d'améliorer l'échantillon test au dessus de 100 estimateurs. L'erreur de l'échantillon d'entraînement continue cependant à baisser ce qui pourrait mener à du sur-apprentissage si le choix du nombre d'estimateurs est trop grand.

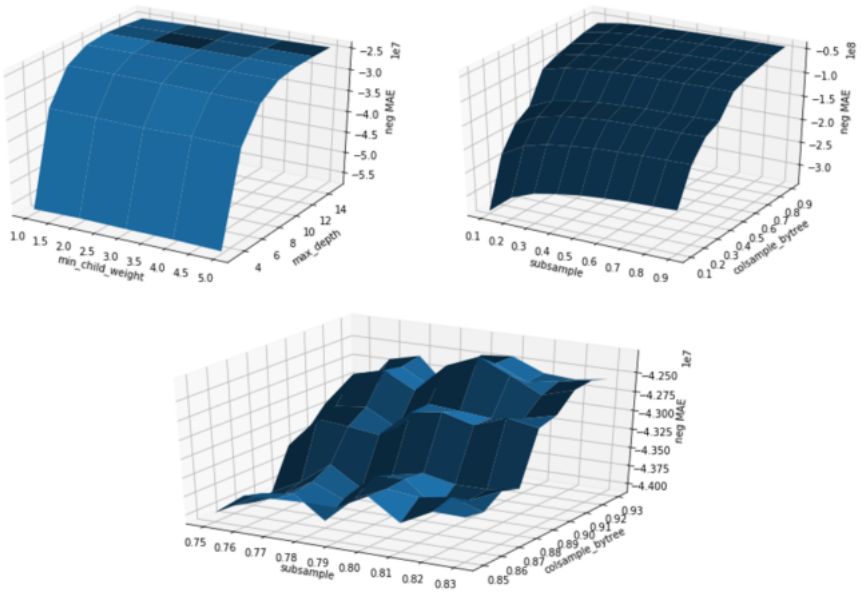



FIGURE 32 – Optimisation des variables du XGB par grid-search réduit

Pour la paire (*max_depth*,*min_child_weight*), seul le premier paramètre a un réel impact sur l'erreur du modèle, avec un score se stabilisant pour une profondeur à partir de 12.

Pour la seconde paire, l'effet est moins unilatéral bien que *colsample_bytree* exerce quand même l'effet le plus fort. Ces paramètres étant des pourcentages, une seconde optimisation sur un intervalle réduit a été effectuée afin de déterminer de manière précise la paire de paramètres optimale.

Les paramètres optimaux obtenus par tuning itératif sont les suivants :

— *max_depth* : 11
— *min_child_weight* : 2
— *learning_rate* : 0.05
— *gamma* : 0.0
— *colsample_bytree* : 0.91
— *n_estimators* : 450
— *subsample* : 0.8
— *reg_alpha* : 50

Les paramètres trouvés sont alors assez différents de ceux obtenus par grid-search. En effet, pour les paramètres optimisés du grid-search, la profondeur est deux fois moins élevée alors que le nombre d'estimateurs est au contraire deux fois plus important. La profondeur de l'arbre

dicte la complexité des arbres, cette plus faible valeur peut venir du fait que la base de données utilisée est plus petite, et ne nécessite (ou ne permet pas) d'avoir des arbres complexes.

Pour les paramètres restés fixes pour le grid-search, les paramètres sont relativement proches des paramètres trouvés pour le tuning itératif, à l'exception du coefficient de régularisation $\alpha$ mais qui n'a de toute façon pas d'impact sur l'erreur si l'on se tient aux graphiques de la figure 31.

**Paramétrage par avis d'expert**

Pour les modèles XGBoost, un modèle avec paramétrage par avis d'expert a aussi été pris en compte. Une des raisons est la limitation que porte l'utilisation d'une base de données réduite pour l'optimisation des hyperparamètres du modèle. Cette problématique sera discutée plus en détail dans la partie [REFERENCE] Limitations du mémoire. Contrairement au Random Forest, le XGBoost est relativement plus rapide à entraîner, et ce même sur une base complète (le temps d'entraînement reste tout de même dans l'ordre des dizaines d'heures). Cela a permis d'effectuer des tests avec des paramètres qui n'auraient pas eu sens sur la base de données réduite, notamment en utilisant des profondeurs plus grandes et un nombre d'estimateurs plus élevé et finalement obtenir un modèle ayant des résultats très satisfaisants.

Le modèle ainsi conçu arbore les paramètres suivants :

— *n_estimators* : 850

— *max_depth* : 25

— *learning_rate* : 0.01

La vitesse d'apprentissage est alors beaucoup plus faible que ceux utilisés pour les autres modèles, alors que le nombre d'estimateurs et la profondeur maximale sont quasiment doublés par rapport au modèle itératif. Le modèle optimisé par grid-search a quant à lui plus d'estimateurs mais une profondeur beaucoup plus faible due à la taille de la base de données utilisée pour optimiser ces paramètres.

## 3.4 Réduction de la dimension et sélection de variables

Un des principaux problèmes évoqués précédemment est l'incapacité d'utiliser la totalité de la base de données à cause de limites dans la puissance de calcul disponible. Cela est dû non pas seulement à cause du grand nombre de scénarios différents (et donc de lignes de tableau différentes) mais aussi à cause du nombre significatif de variables explicatives. De plus, l'effet sur la taille de la base de données est d'autant plus important que retirer une ligne du tableau va alléger le tableau d'environ 140 valeurs, tandis que retirer une variable explicative, c'est à dire une colonne du tableau, va en retirer plus de 700 000.

La réduction de la dimension de la base de données et donc la sélection des variables explicatives sont donc des enjeux cruciaux pour l'amélioration des performances futures des modèles.

Avant même de s'intéresser aux algorithmes permettant la réduction de variables, il est néces-

saire de décider si la base de données entière sera utilisée ou non pour cette sélection. L'utilisation de la base de données complète permettra d'avoir un résultat plus pertinent lors des tests de prédiction réels mais induira potentiellement un résultat moins pertinent pour les modèles utilisant des bases de données réduites. Le but étant d'utiliser à terme la plus grande base de données possible, les réductions de variables seront conduites sur la base totale.

Plusieurs pistes seront ici explorées avec divers techniques et algorithmes de sélection automatique de variables.

### 3.4.1 Régression LASSO

La régression LASSO (Least Absolute Shrinkage and Selection Operator) est une méthode qui permet d'effectuer à la fois de la sélection de variables et de la régularisation (processus permettant de "simplifier" le résultat), tout cela dans le but d'augmenter la justesse des prédictions d'un modèle.

Cette méthode fonctionne en sélectionnant les variables qui vont minimiser une certaine formule sous contrainte.

Plus précisément, soit $x_i$ le vecteur contenant les variables explicatives associées à un scénario $i$ et $y_i$ la réponse correspondante. $\beta$ représente quant à lui le vecteur des coefficients que la régression LASSO cherche à estimer, et qui peut être vu comme un genre "d'utilité" de la variable explicative pour la prédiction de la variable réponse.

Les coefficients $\beta$ sont alors obtenus via une minimisation de la somme des carrés des résidus, c'est à dire :

$$\min_{\beta_1,\beta_2,\ldots,\beta_M} \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{M}\beta_j x_{i,j}\right)^2 + \alpha\sum_{j=1}^{M}|\beta_j| \tag{11}$$

Avec $\alpha \geq 0$.

Le coefficient $\alpha$ permet alors de contrôler le niveau de régularisation des coefficients que l'on estime. Plus ce coefficient est grand, plus les variables explicatives peu pertinentes sont "forcées" à 0 pour effectuer le plus grand gain de minimisation tout en restant cohérent vis à vis de la somme des résidus au carré.

La sélection de cet hyperparamètre est effectuée de la même manière que les précédents en parcourant une plage de valeur potentielle et en sélectionnant la valeur du paramètre minimisant l'erreur choisie. La plage ici choisie va de 0.1 à 1 par incrément de 0.1.

A l'issue de cette régression, il suffit d'analyser les coefficients retournés et ne garder que les variables ayant un coefficient estimé différent de 0.

Seules deux variables ont été retenues par cette opération :

— **CFP_VAN** qui correspond à la valeur actualisée nette des coûts en fonds propre de l'entreprise

— **PNA_VAN** qui correspond à la valeur actualisée nette des produits nets d'assurances

Les Coûts de Fonds Propres sont utilisés pour le calcul des produits nettes d'assurance, qui est à son tour utilisé lors du calcul de l'Equity. Leur sélection est alors facilement explicable étant donné que ces variables jouent un rôle important dans l'évaluation de la variable réponse dans le cas d'un déroulement de la formule standard usuelle.

Bien que le nombre de variables ait été divisé par plus de 70, la puissance de calcul disponible ne permet pas d'entraîner le modèle sur la base de données initiale dans des temps convenables. La proportion de la base utilisée sera donc de **20%** Les paramètres obtenus en utilisant uniquement ces variables pour les modèles Random Forest et XGBoost sont les suivants :

Random Forest :

— *max_depth* : 10

— *min_samples_split* : 2

— *n_estimators* : 1000

XGBoost :

— *max_depth* : 8

— *learning_rate* : 0.01

— *n_estimators* : 700

Les paramètres obtenus diffèrent assez fortement des paramètres obtenus initialement : La profondeur est bien moins élevée et le nombre d'estimateurs a presque doublé pour le Random Forest. Cette réduction de la profondeur est très certainement liée au fait du nombre réduit de variable explicative. Cependant avec aussi peu de variables, la variance entre les arbres sera assez élevée, d'où la nécessité d'augmenter le nombre d'estimateurs afin de réduire la variance totale de la forêt.

Dans le cas du XGBoost, la plus grande différence se situe aussi sur le nombre d'estimateurs, qui est ici plus faible. Il est intéressant de remarquer que la vitesse d'apprentissage est aussi plus faible, alors que ce paramètre est généralement inversement proportionnel au nombre d'estimateurs : plus le modèle apprend lentement, plus il lui faut d'estimateurs. La réduction du nombre de variables permet alors d'avoir une vitesse d'apprentissage plus lente tout en réduisant le nombre d'estimateurs grâce à la baisse de la complexité des données.

En plus d'utiliser la régression LASSO comme une technique de réduction de variables, cette régression a été utilisée comme modèle de prédiction pour la suite afin d'avoir une comparaison avec un modèle simple.

Le modèle a donc été optimisé sur **la base complète**, en faisant varier son seul paramètre $\alpha$ entre 0.1 et 10, par pas de 0.1 .

Le modèle LASSO optimal obtenu est donc alors pour $\alpha = 0.1$

### 3.4.2 Recursive Feature Elimination (RFE)

Le Recursive Feature Elimination est un algorithme permettant de sélectionner les variables les plus pertinentes. Cette méthode fonctionne en entraînant le modèle sur l'ensemble des

variables explicatives, puis en éliminant la ou les variables les plus faibles. Ce processus est alors répété autant de fois que nécessaire jusqu'à obtenir le nombre de variables souhaité.

L'algorithme fonctionne alors de la manière suivante :

1. Entraînement du modèle
2. Calcul du rangs de toutes les variables explicatives
3. Suppression de la variable la plus faible (ayant le rang le plus faible)

A chaque itération $i$, un nouveau sous ensemble de variable $F_i$ est crée en ne gardant que les variables les plus fortes, avec $F_i \subset F_{i-1} \subset ... \subset F_0$. Il faut cependant prendre en compte que les variables les mieux classées ne sont pas forcément les plus significatives individuellement. Le classement ne prend sens que si l'ensemble des variables retenues n'est considéré et le sous ensemble $F_i$ n'est "optimal" que de cette manière. Si l'on devait comparer individuellement la performance de chaque variable, alors l'ensemble de même taille obtenu en ne retenant que les variables les plus significatives serait potentiellement différent.

Le critère de rang utilisé dans le cadre de ce mémoire va dépendre du modèle pris en compte. Les modèles considérés ici étant le Random Forest et le XGBoost, deux critères de rang seront donc utilisés pour déterminer les jeux de paramètres optimaux.

Les deux modèles utilisent une même mesure appelée "feature importance". Cette mesure permet de calculer l'importance de chaque variable afin de les classer et déterminer les variables les plus significatives pour le modèle. Elle correspond à la **réduction de l'impureté du noeud, pondérée par la probabilité d'atteindre ce noeud**. L'impureté mentionnée ici correspond à une mesure d'erreur et qui est dans notre cas le MAE.

Pour chaque noeud $j$, l'importance de ce noeud est donnée par :

$$ni_j = w_j C_j - w_{left}(j) C_{left}(j) - w_{right}(j) C_{right}(j) \tag{12}$$

avec

— $ni_j$ l'importance du noeud $j$

— $w_j$ la proportion d'échantillons atteignant le noeud $j$

— $C_j$ l'impureté liée au noeud $j$

— $left(j)$ Le noeud fils de gauche du noeud $j$

— $right(j)$ Le noeud fils de droite du noeud $j$

L'importance de chaque variable explicative $i$ est ensuite calculée de la manière suivante :

$$fi_i = \frac{\sum_{ni_j \in F_i} ni_j}{\sum_{ni_k \in N} ni_k} \tag{13}$$

avec :

— $fi_i$ l'importance de la variable explicative $i$

— $F_i$ l'ensemble des noeuds utilisant la variable $i$ comme critère de division

— $N$ l'ensemble des noeuds de l'arbre

L'importance de la variable est alors normalisée :

$$fi'_i = \frac{fi_i}{\sum_{j \in AllFeatures} fi_j}$$

et finalement, pour obtenir l'importance de la variable explicative pour la forêt entière, il suffit de prendre la moyenne de l'importance de la feature à travers les arbres :

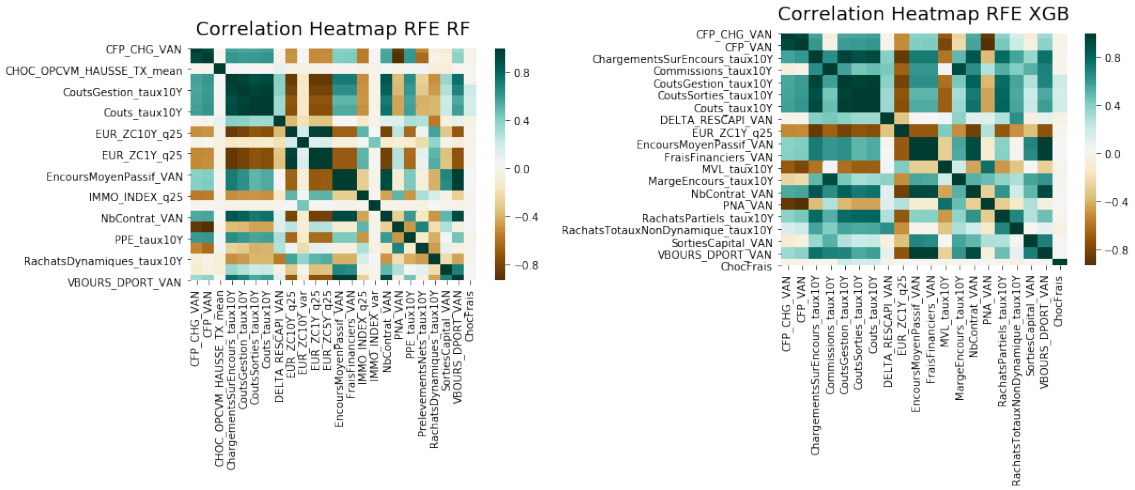
$$RF(fi_i) = \frac{\sum_{j \in arbres} fi'_{i,j}}{T}$$

avec

— $fi'_{i,j}$ l'importance de la variable $i$ pour l'arbre $j$ normalisée

— $T$ le nombre d'arbres de la forêt

Afin de pouvoir utiliser cet algorithme, il est nécessaire de déterminer le nombre de variables explicatives à garder. Ce nombre a été fixé à 24 afin de ne pas retirer les variables dont les effets n'auront pas été potentiellement suffisamment significatifs à cause de l'utilisation d'un extrait de la base de données.




(a) RandomForest utilisé comme estimateur

(b) XGBoost utilisé comme estimateur

FIGURE 33 – Heatmap des corrélations inter-variables après RFE pour les deux modèles

En comparant avec la heatmap obtenue avant RFE à la figure 17, il est clair que cette opération à retiré de nombreuses variables dont la plupart ayant une faible corrélation entre-elles. Cette réduction de la corrélation inter-variables n'est pas un but en soi du RFE mais peut cependant traduire une tendance que les variables explicatives les plus corrélées entre elles soient les plus importantes. Il est possible que la corrélation entre ces variables et la variable cible soit forte, et donc en retenant ces dernières, elles seraient alors corrélées entre elles. Au contraire, les

variables peu corrélées avec les autres ne seraient alors aussi que peu corrélées ou importantes pour le modèle et la variable cible. Il est difficile de prouver cette hypothèse mais la première peut être illustrée en regardant la corrélation entre les variables retenues et la variable cible.

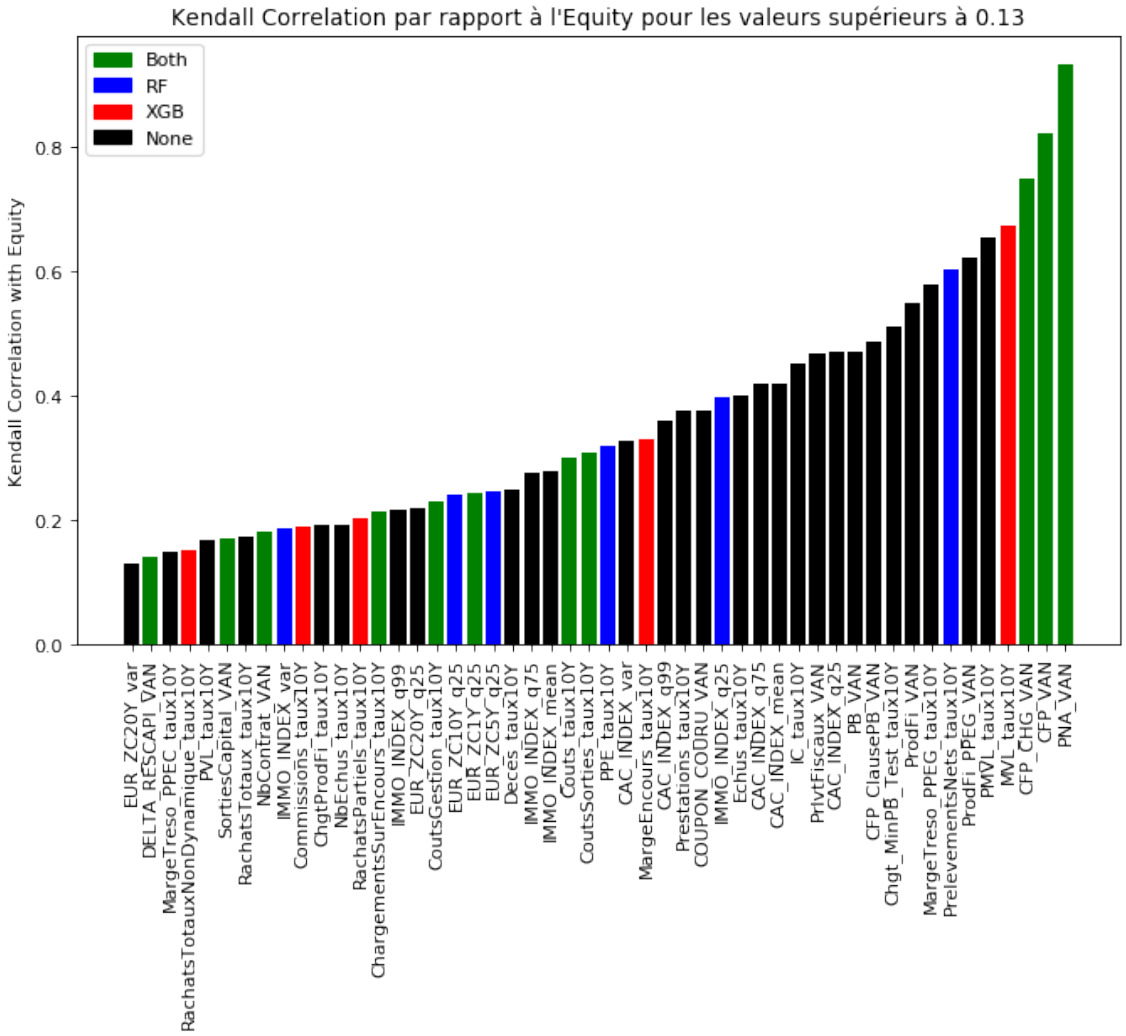



FIGURE 34 – Comparaison de la corrélation entre les variables explicatives et l'Equity et de leurs présences dans les variables séléctionnées par RFE

Ce graphique montre les variables explicatives les plus corrélées avec la variable réponse, ainsi que leur appartenance ou non aux variables sélectionnées par RFE en utilisant le XGBoost ou le RF comme estimateur. Les variables ayant une corrélation faible (<0.13) ont été retirées afin de ne pas polluer inutilement le graphique. Les barres vertes correspondent aux variables retenues par le XGBoost et le RF, les bleues celles retenues par le RF, les rouges par le XGBoost et les noires par aucun des deux.

Cela confirme alors l'hypothèse que la corrélation joue un rôle dans le choix du RFE de garder ou non une variable. Les trois variables les plus corrélées avec l'Equity ont été gardées par à la fois par le XGBoost ainsi que le RF et il est intéressant de remarquer que les deux variables les plus corrélées sont celles retenues par la régression LASSO.

Cependant, la corrélation n'est pas l'unique facteur à prendre en compte comme le montrent

les nombreuses variables non sélectionnées qui ont pourtant un coefficient de corrélation supérieur à 0.4 .

Bien que le nombre de variables soit réduit par rapport à l'ensemble initial (près de 6 fois moins de variables), la problématique reste identique à celle rencontrée avec la régression LASSO et une optimisation sur le jeu de données initial reste impossible. L'optimisation sera donc effectuée sur **10%** de la base de données afin de quand même profiter de cette réduction de dimension.

Suite à cette sélection de variables, une nouvelle optimisation par grid-search a été effectuée sur les mêmes plages de paramètres et les jeux de paramètres optimaux obtenus sont les suivants :

Pour le Random Forest on a :

- *max_depth* : 150
- *min_samples_split* : 2
- *n_estimators* : 600

Pour le XGBoost :

- *max_depth* : 6
- *learning_rate* : 0.05
- *n_estimators* : 1150

Contrairement aux paramètres obtenus après la régression LASSO, les hyperparamètres post RFE sont beaucoup plus proches des paramètres initiaux, avec seulement de légères différences sur la profondeur du Random Forest et le nombre d'estimateurs et vitesse d'apprentissage du XGBoost. Le fait d'avoir gardé plus de 20 variables a permis au modèle de capter la plupart des informations utilisées par le modèle originel et donc d'être similaires en termes de paramètres après optimisation. Le fait d'avoir des paramètres similaires peut néanmoins mener à des résultats assez différents, notamment sur les classes de données les moins représentées.

Maintenant que sont définis plusieurs modèles, jeux de paramètres et jeux de variables différents, il est alors possible de les comparer les uns aux autres afin de déterminer le ou les modèles les plus performants.

# 4 Résultats, interprétation et critiques

Dans cette partie, plusieurs comparaisons des modèles utilisant des métriques différentes seront effectuées dans le but de déterminer le ou les modèles les plus performants et donc de conclure sur la pertinence des données utilisées, des modèles, ainsi que de l'utilité opérationnelle de ces travaux.

Dans la suite, les modèles seront nommés de la manière suivante :

— Modèle PRE : Modèle optimisé par grid-search avec toutes variables (pre-RFE)

— Modèle POST : Modèle optimisé par grid-search avec les variables du RFE (post-RFE)

— Modèle ITER : Modèle optimisé de manière iterative

— Modèle EXPERT : Modèle paramétré par avis d'expert

Les modèles étudiés sont les suivants :

— Lasso

— Random Forest :

  · Toutes variables non optimisé (RF baseline)

  · Toutes variables et optimisé par grid-search (RF pre)

  · Toutes variables et optimisé de manière itérative (RF iter)

  · Variables LASSO et optimisé par grid-search (RF Lasso)

  · Variables RFE et optimisé par grid-search (RF post)

— XGBoost :

  · Toutes variables non optimisé (XGB baseline)

  · Toutes variables et optimisé par grid-search (XGB pre)

  · Toutes variables et optimisé de manière itérative (XGB iter)

  · Variables LASSO et optimisé par grid-search (XGB Lasso)

  · Variables RFE et optimisé par grid-search (XGB post)

  · Toutes variables et paramètres par avis d'expert (XGB expert)

Avant d'effectuer la comparaison détaillée des résultats, il peut être intéressant de regarder l'erreur absolue moyenne (MAE) de chaque modèle pour la base de données d'entraînement entière.

Le MAE et le Root Mean Squared Error (RMSE, Erreur quadratique moyenne) sont les deux métriques qui seront utilisées par la suite.

Elles sont définies de la manière suivante :

$$MAE(y) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{14}$$

avec $y$ la variable à prédire (ici l'Equity), $y_i$ et $\hat{y}_i$ la i-ème observation et prédiction.

$$RMSE(y) =$$

Le modèle Random Forest avec tuning itératif n'a pas été représenté car l'erreur issue de ce modèle était bien plus important que les autres avec une erreur de l'ordre de 600 millions. Il a donc été retiré pour éviter de rendre difficile la lecture du graphique.

Dans l'ensemble, la plupart des modèles donnent de plutôt bons résultats, avec un MAE qui se situe sous les 30 millions. En comparant avec la grandeur des valeurs prédites qui est en milliards, les résultats peuvent même être considérés comme très satisfaisants.

On peut dès lors éliminer les modèles les moins performants, c'est à dire le LASSO, le XGBoost de base et le Random Forest utilisant les variables LASSO.

On peut aussi remarquer que par rapport au modèle de base, assez peu d'améliorations ont été apportées au Random Forest. Cela est probablement dû partiellement à l'optimisation incomplète de ce dernier, contrairement au XGBoost qui a eu droit à un paramétrage par avis d'expert qui est notre modèle le plus performant à première vue et qui est par rapport au MAE global, près de 8 fois plus performant que le modèle de base.

Afin de limiter les modèles considérés n'apportant que peu d'améliorations de performance, les modèles retenus pour l'étude des résultats seront les suivants :

— XGB optimisé par grid-search
— XGB optimisé de manière itérative
— XGB paramétré par avis d'expert
— Random Forest optimisé par grid-search
— Random Forest optimisé par grid-search avec variables RFE.

Les modèles Random Forest optimisés par grid-search avant et après RFE ont été retenus car la différence en performance n'est pas visible facilement depuis la figure **??**. Une étude plus détaillée est alors nécessaire pour les distinguer.

## 4.1 Études des résultats

Les modèles utilisés peuvent mener à des erreurs absolues moyennes plus ou moins proches. Cependant cette mesure est imparfaite car donne le même poids aux grands écarts qu'aux petits, et des compensations peuvent donc être effectuées entre les erreurs des différents chocs.

Afin de mieux comprendre les modèles retenus, une étude **choc par choc** sera effectuée afin de pouvoir évaluer les modèles par choc et déterminer le plus performant pour un choc spécifique.

Le MAE et le RMSE global de ces modèles sont les suivants :

Le modèle étant globalement le plus performant est alors le XGBoost avec paramètres par avis d'expert, avec un MAE près de **30%** inférieur et un RMSE plus de **10%** plus bas.

Afin de comparer les différents modèles retenus, une étude du MAE choc par choc à d'abord été effectuée.

| | XGB Pre | XGB Iter | XGB Expert | RF Pre | RF Post |
|---|---|---|---|---|---|
| MAE | 2.09e+07 | 1.33e+07 | 9.62e+06 | 1.55e+07 | 1.55e+07 |
| RMSE | 2.85e+07 | 2.08e+07 | 1.85e+07 | 2.99e+07 | 2.92e+07 |

TABLE 4 – MAE et RMSE des différents modèles XGBoost

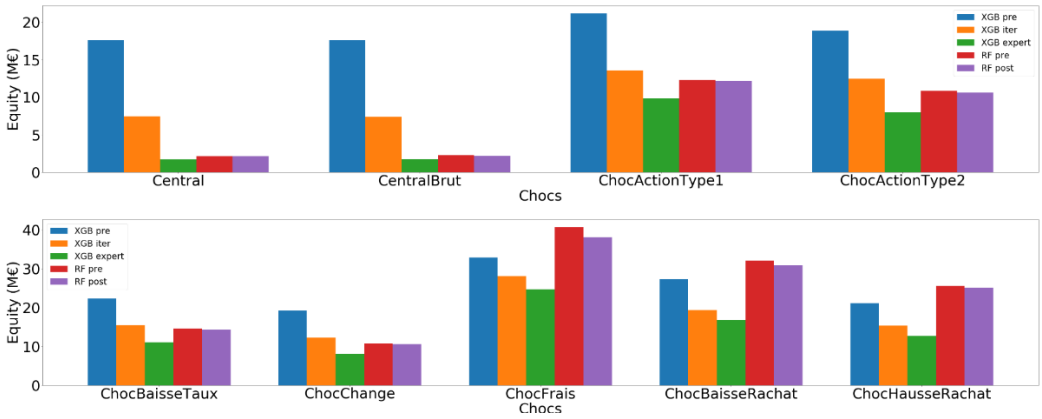



FIGURE 35 – Comparaison du MAE des modèles par choc (part 1)

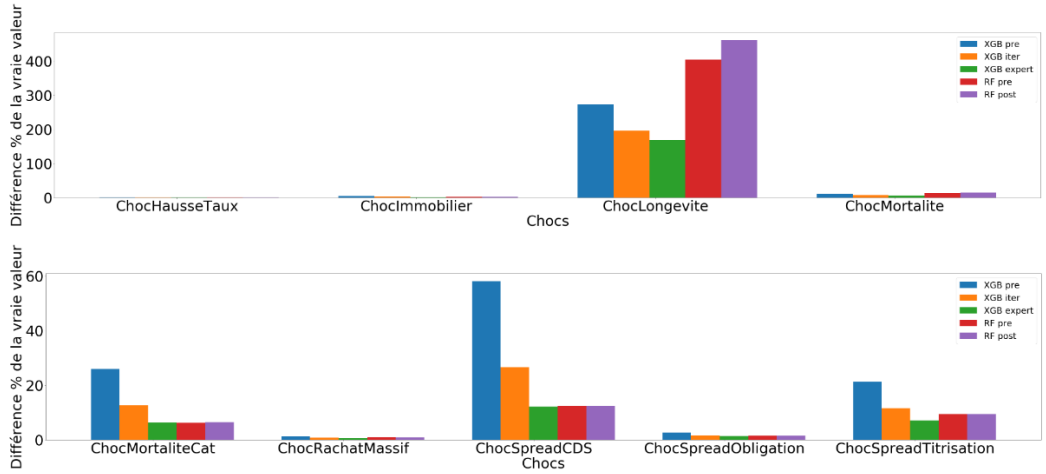



FIGURE 36 – Comparaison du MAE des modèles par choc (part 2)

Pour les figures suivantes, l'erreur relative a été retenue comme mesure afin d'avoir une visualisation plus facile des performances des différents modèles en fonction du choc appliqué. Bien que certains modèles aient des erreurs assez élevées pour certains chocs, dans leur globalité, les modèles performent de manière assez satisfaisante avec des erreurs étant régulièrement en dessous des 10 millions d'euros. Cependant certains chocs ont une erreur bien plus importante que les autres, et ce, peu importe le modèle en question. Il est alors intéressant d'étudier non pas les erreurs moyennes absolues de manière brute, mais de les rapporter à la variable cible moyenne absolue afin d'apprécier l'erreur de manière relative.

Pour un choc $Choc$, l'erreur relative obtenue est alors calculée de la manière suivante :

$$ErrRel_{Choc} = \frac{MAE_{Choc} - |MoyenneReelle_{Choc}|}{|MoyenneReelle_{Choc}|}$$

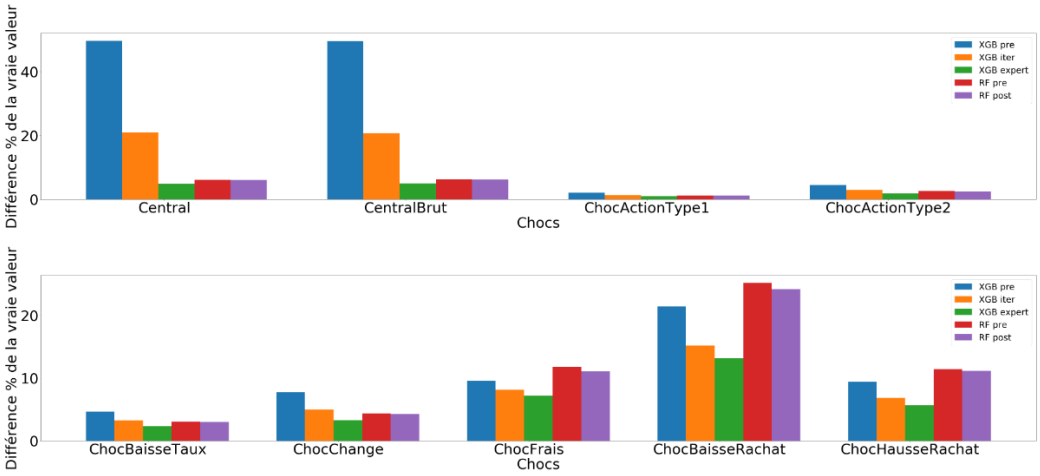



FIGURE 37 – Comparaison des modèles en erreur relative par choc (part 1)

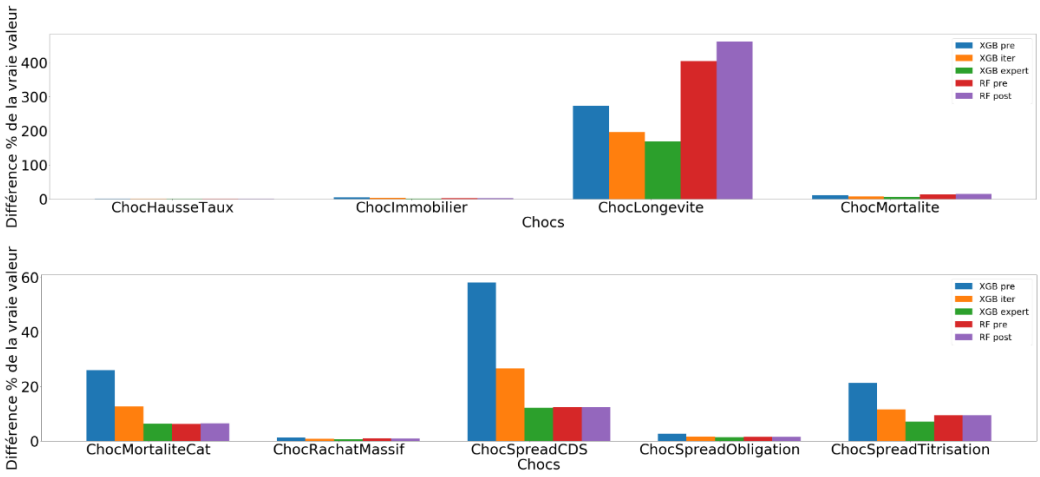



FIGURE 38 – Comparaison des modèles en erreur relative par choc (part 2)

Dans les figures 37 et 38, certains modèles se distinguent rapidement par une erreur relative régulièrement plus élevée que les autres. En effet, pour le modèle XGB PRE, l'erreur pour les scénarios *Central* et *CentralBrut* (qui correspond au montant central brut de réassurance) sont clairement plus élevée que l'ensemble des autres modèles, et ne sera alors pas retenu. Une même analyse peut être faite pour les modèles se basant sur les Random Forest avec des erreurs relatives très importantes sur les chocs Rachat et Longévité. Cependant, il n'est pas possible de les éliminer tout de suite pour autant car ce sont les modèles les plus performants pour de nombreux autres chocs tels que *ChocMortalitéCat, ChocSpreadCDS* ou encore *ChocActionType1*, bien que suivi de très près par d'autres modèles.

Le modèle XGB Iter est performant sur tous les chocs mais n'est que rarement le meilleur modèle, contrairement au XGB expert qui a aussi une erreur très faible sur l'ensemble des chocs, mais est en plus très souvent le meilleur modèle pour un choc donné. On a donc que bien que les MAE et RMSE soient proches, des différences marquantes entre les différents modèles.

Les modèles RF Pre et RF post étant très similaires, le modèle RF Pre sera retenu pour la suite

afin de ne pas surcharger les graphiques et les résultats.

En regardant de plus près sur les différents modèles retenus il est possible de déterminer pour chaque choc, le modèle le plus performant.

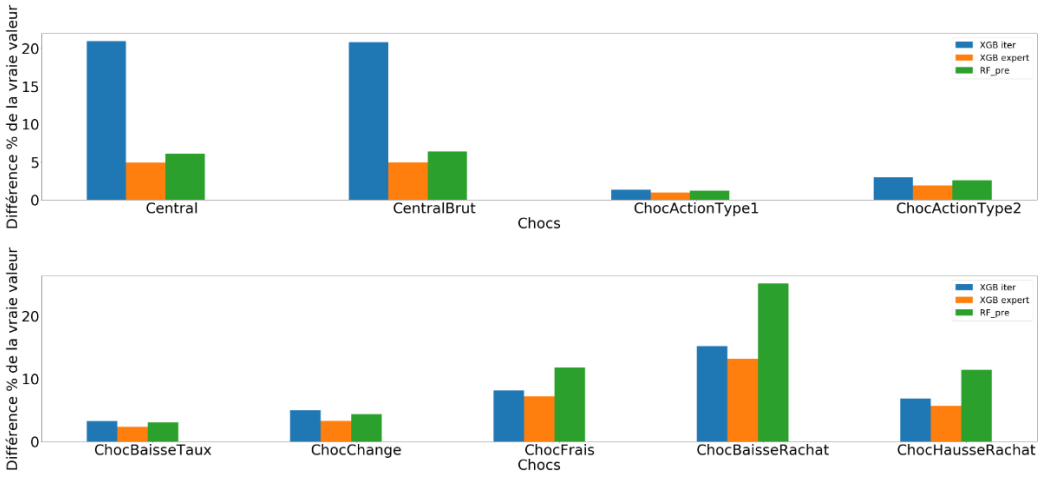



FIGURE 39 – Comparaison des modèles retenus en erreur relative par choc(part 1)

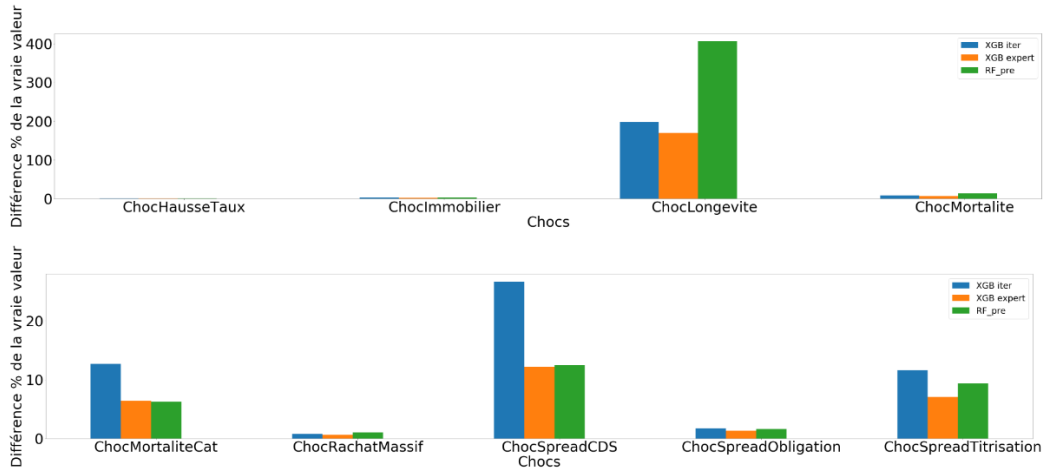



FIGURE 40 – Comparaison des modèles retenus en erreur relative par choc (part 2)

Les modèle les plus optimaux par choc est donc les suivants :

— Central : **XGB Expert**

— ChocActionType1 : **XGB Expert**

— ChocActionType2 : **XGB Expert**

— ChocBaisseTaux : **XGB Expert**

— ChocChange : **XGB Expert**

— ChocFrais : **XGB Expert**

— ChocBaisseRachat : **XGB Expert**

— ChocHausseRachat : **XGB Expert**

— ChocHausseTaux : **XGB Expert**

— ChocImmobilier : **XGB Expert**

— ChocLongevite : **XGB Expert**

— ChocMortalite : **XGB Expert**

— ChocMortaliteCat : **XGB Expert**

— ChocRachatMassif : **XGB Expert**

— ChocSpreadCDS : **RF Pre**

— ChocSpreadObligation : **XGB Expert**

— ChocSpreadTitrisation : **XGB Expert**

Le choc CentralBrut n'est pas considéré car n'est pas un choc réglementaire requis par Solvabilité 2.

Les modèles les plus performants sont alors :

— Le modèle XGB Expert pour 16 chocs

— Le modèle RF Pre pour un unique choc

Le modèle XGB Iter n'a été retenu pour aucun des chocs, tandis que le modèle XGB Expert est le plus performant sur la quasi-totalité des chocs. Seul le choc MortalitéCat s'est vu être plus performant avec le modèle RF Pre bien que la différence avec le modèle phare ne soit que très minime.

Le choc Longevite a quant à lui une erreur très importante, peu importe le modèle choisi (plus de 150% d'erreur au minimum). Ce résultat est soit dû aux mauvaises performances des modèles, soit au fait que les valeurs en jeu soient faibles et que donc les erreurs relatives explosent rapidement.

Les prédictions des chocs longévités sont nettement moins importants que certains autres chocs ce qui expliquerait alors les erreurs relatives : les ordres de grandeur du choc Longévité sont de **1e10** tandis que que les autres chocs sont de l'ordre de **1e12** ou **1e13**. Cela n'explique évidemment pas complètement l'erreur qui provient aussi en grande partie de la faible performance des modèles sur ce choc.

En plus de savoir si le modèle est performant, il est intéressant de regarder si le modèle reproduit la distribution historique de l'Equity.

## 4.2 Étude de la distribution de la variable cible et des prédictions

Dans une précédente partie, l'hypothèse avait été émise que la variable Equity, une fois soumise à certaines transformations, pouvait potentiellement être une loi Gamma.

Les deux lois qui seront testées sont les lois **Gamma** et les lois **Beta**.

### 4.2.1 Comparaison avec des distributions existantes

#### Transformation de la variable cible

Les lois Gamma et lois Beta sont utilisées sur des supports différents que les valeurs atteintes par la variable cible.

La loi Gamma est une loi **positive** avec son pic de probabilité qui est situé du côté de zéro : il est alors nécessaire de prendre la valeur opposée de l'Equity ainsi que de rendre l'ensemble positifs.

$$Y_{gamma} = -Y + |min(Y)|$$

Dans le cas de la loi Beta, les valeurs sont en plus d'être positives, normalisées entre 0 et 1. Donc, en effectuant la même inversion pour garder le pic de probabilité du bon côté :

$$Y_{beta} = \frac{-Y - min(-Y)}{max(-Y) - min(-Y)}$$

ce qui est simplement une normalisation du type *MinMax* pour la variable $-Y$.

Afin de pouvoir comparer les valeurs empiriques avec ces distributions, il est nécessaire de déterminer les différents paramètres qui vont être utilisés pour ces lois.

**Détermination des paramètres des lois de probabilité par méthode des moments**

La méthode des moments est une méthode permettant d'estimer des paramètres recherchés en égalisant des moments théoriques (et qui dépendent de ces paramètres).

L'approximation des moments théoriques, généralement la moyenne et la variance, est justifiée par la loi des grands nombres qui permet d'obtenir des estimateurs de ces moments théoriques.

Soient $x_1, ..., x_n$ un échantillon i.i.d d'une distribution paramétrique paramétrée par $\theta$.

Chaque moment $m_i$ peut être approché par $\hat{m}_i$ avec :

$$\hat{m}_i = \frac{1}{n} \sum_{k=1}^{n} x_k^i$$

Si on considère $s$ moments $G = [m_1(\theta), ..., m_s(\theta)]$ et son estimation empirique $\hat{G}$, l'estimation par méthode des moments revient à résoudre $\hat{G} = G(\hat{\theta})$

Dans le cas de la loi Gamma, les deux paramètres à estimer sont : $\alpha$ et $\beta$.

On a les égalités suivantes :

$$E[X] = m_1 = k\theta$$
$$E[X^2] = m_2 = \theta^2 k(k+1)$$

Ayant deux équations avec deux inconnus, le système peut être résolu et :

$$k = \frac{m_1^2}{m_2 - m_1^2}$$
$$\theta = \frac{m_2 - m_1^2}{m_1}$$

en remplaçant $m_1$ et $m_2$ par leurs estimations empiriques :

$$\hat{m_1} = \frac{x_1 + ... + x_n}{n}$$
$$\hat{m_2} = \frac{x_1^2 + ... + x_n^2}{n}$$

Les paramètres estimés sont alors :

$$\hat{k} = \frac{\hat{m_1}^2}{\hat{m_2} - \hat{m_1}^2}$$
$$\hat{\theta} = \frac{\hat{m_2} - \hat{m_1}^2}{\hat{m_1}}$$

Dans le cas de la loi Beta, les équations sont les suivantes :

$$E[X] = \frac{\alpha}{\alpha + \beta}$$
$$E[X^2] - E[X]^2 = \sigma^2 = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}$$

En posant $k = \frac{1-E[X]}{E[X]}$, on a :

$$\alpha = \frac{k - \sigma^2(1+k)^2}{\sigma^2(1+k)^3}$$
$$\beta = k\alpha$$

Les paramètres ainsi obtenus sont les suivants :

| Loi Gamma | Equity | RF Pre | XGB Iter | XGB Expert |
|---|---|---|---|---|
| $k$ | 4.81 | 4.57 | 4.79 | 4.77 |
| $\theta$ | 1.92e+09 | 1.97e+09 | 1.92e+09 | 1.93e+09 |

TABLE 5 – Paramètres de la loi Gamma par méthode des moments

| Loi Beta | Equity | RF Pre | XGB Iter | XGB Expert |
|---|---|---|---|---|
| $\alpha$ | 3.58 | 3.42 | 3.59 | 3.57 |
| $\beta$ | 13.38 | 13.22 | 13.67 | 13.93 |

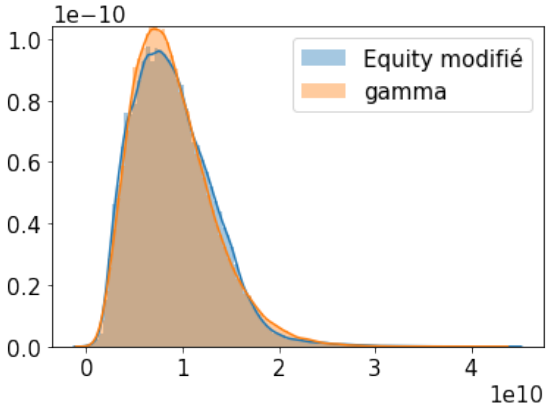
TABLE 6 – Paramètres de la loi Beta par méthode des moments

Le meilleur cas serait alors que les conditions suivantes soient remplies :

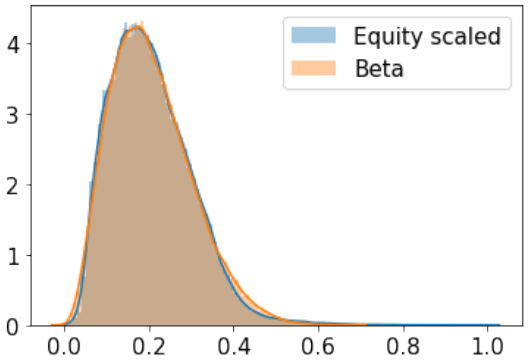
— Il est statistiquement probable que la variable cible suive une des deux lois testées

— Les paramètres de la loi en question sont proches ou égaux de ceux obtenus en utilisant les données prédites du modèle optimal

Dans le cas où la variable cible ne suive pas une de ces deux lois, il est quand même possible de comparer cette dernière avec nos modèles et ainsi déterminer s'ils suivent une même distribution inconnue.




(a) Comparaison de l'Equity et de la loi Gamma obtenue par méthode des moments




(b) Comparaison de l'Equity et de la loi Beta obtenue par méthode des moments

FIGURE 41 – Étude visuelle des lois de probabilités potentielles

A première vue, la loi Beta est beaucoup plus proche et adaptée à représenter la loi de la variable cible. Un détachement peut cependant être repéré autour de $x = 0.03$ et autour de $x = 0.4$ . Afin de pouvoir déterminer le "goodness of fit" de ces lois (leur capacité à représenter correctement nos données), plusieurs tests statistiques existent et permettent alors de donner une probabilité que les deux échantillons proviennent de la même distribution.

Les tests suivants seront effectués :

— Test de Kolmogorov–Smirnov

— Critère de Cramér–von Mises

### 4.2.2 Kolmogorov Smirnov Test

Le premier test qui sera effectué sera le test de Kolmogorov Smirnov. C'est un test non paramétrique (c'est à dire qui ne se base pas uniquement sur des familles paramétrées de distribution de probabilité) qui permet de comparer un échantillon avec une loi de probabilité (one-sample) ou un autre échantillon (two-samples), et de déterminer si ils dérivent de la même distribution.

**La statistique de Kolmogorov-Smirnov**

La fonction de distribution empirique d'un échantillon de taille $n$ est définie de la manière suivante, avec $(X_i)$ les observations triées par ordre croissant :

$$F_n(x) = \frac{\text{Nombre d'éléments dans l'échantillon } \leq x}{n} = \frac{1}{n}\sum_{i=1}^{n} 1_{[-\infty,x]}(X_i)$$

On en déduit alors la statistique par :

$$D_n = \sup_x |F_n(x) - F(x)| \tag{16}$$

La statistique est donc une mesure de l'écart maximale entre la fonction de distribution empirique, et la loi à laquelle elle est comparée (ou à la seconde distribution empirique).

La statistique est rejetée pour le seuil $\alpha \in [0,1]$ dans le cas où :

— Pour le one-sample test : $\sqrt{n}D_n > K_\alpha$

— Pour le two-samples test : $D_{n,m} > c(\alpha)\sqrt{\frac{n+m}{n\times m}}$

avec

$$K_\alpha = P(K \leq K_\alpha) = 1 - \alpha$$

$$K = \sup_{t\in[0,1]} |B(t)|$$

$B(t) = W_t - tW_1$ un pont Brownien, avec $W_t$ un mouvement Brownien

$$c_\alpha = \sqrt{-\ln(\alpha/2)/2}$$

Lors de l'utilisation de cette statistique, l'hypothèse testée (hypothèse nulle) est :

$$H_0 = \text{Les deux échantillons suivent la même distribution} \tag{17}$$

Il est alors possible de calculer pour cette statistique et cette hypothèse la p-value associée, qui correspond à la probabilité que $H_0$ soit vrai.

Si cette p-value est en dessous du seuil $\alpha$, alors on peut rejeter l'hypothèse. Dans le cas contraire, on ne peut pas rejeter l'hypothèse $H_0$ mais cela ne veut pas dire que l'hypothèse est vraie pour autant.

Les statistiques et pvalues pour la variable cible et les lois Gamma et Beta considérées sont

alors les suivantes :

| Loi Gamma | Equity | RF Pre | XGB Iter | XGB Expert |
|---|---|---|---|---|
| Statistique | 0.996 | 0.994 | 0.996 | 0.996 |
| p-value | 0.0 | 0.0 | 0.0 | 0.0 |

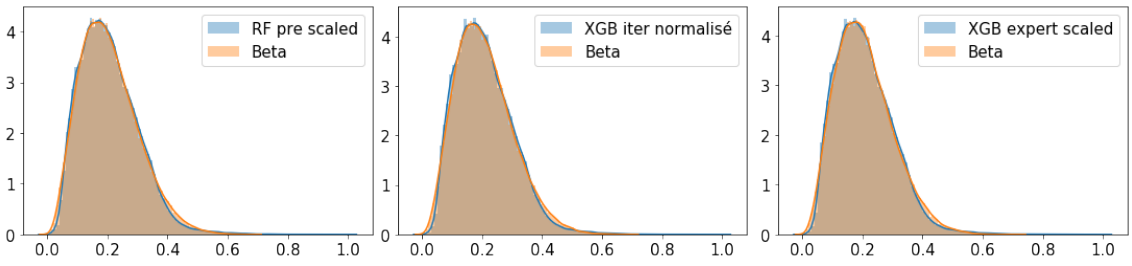
TABLE 7 – Statistiques et p-values pour le test de Kolmogorov-Smirnov avec la loi Gamma

La statistique pour la loi Gamma est totalement rejetée pour l'ensemble de nos modèles. Cette conclusion était prévisible, l'écart maximale entre les deux distributions étant clairement visible même à l'oeil nue.

| Loi Beta | Equity | RF Pre | XGB Iter | XGB Expert |
|---|---|---|---|---|
| Statistique | 0.012 | 0.010 | 0.011 | 0.011 |
| p-value | 4.5e-19 | 6.2e-14 | 4.67e-18 | 1.41e-17 |

TABLE 8 – Statistiques et p-values pour le test de Kolmogorov-Smirnov avec la loi Beta

Pour la loi Beta, l'hypothèse est de même fortement rejetée pour l'ensemble de nos modèles. Cependant ce rejet est moins important que pour la loi Gamma, ce qui correspond à l'adéquation plus forte de la loi aux observations constatées précédemment.




(a) (b) (c)

| | RF Pre | XGB Iter | XGB Expert |
|---|---|---|---|
| Statistique | 3.88e-4 | 3.81e-4 | 3.74e-4 |
| p-value | 1.0 | 1.0 | 1.0 |

(d)

FIGURE 42 – Comparaison des distributions empiriques des modèles avec les observations (a,b,c) ; Statistiques et p-values pour le test de Kolmogorov-Smirnov two-samples (d)

La comparaison des résultats des modèles et des observations donne cependant des conclusions différentes. A l'oeil nu, il est impossible d'apprécier la différence entre les différents modèles. Les trois modèles considérés suivent parfaitement la distribution de la variable cible. Les statistiques n'apportent pas non plus d'information supplémentaire car l'hypothèse $H_0$ ne peut être rejetée pour aucun seuil $\alpha$, l'ensemble des p-values pour le test two-samples étant égales à 1.

Il n'est cependant pas possible de faire confiance au test de Kolmogorov-Smirnov de manière certaine. En effet, une des conditions d'utilisation de ce test est l'indépendance des deux échantillons testés. Dans le cas des distributions de probabilité, les paramètres des lois ont été déduits

des observations, il n'y a alors pas indépendance. Dans le cas des modèles, ces derniers ont été optimisés et entraînés sur une partie des observations. L'indépendance avec les observations de la variable cible est donc aussi discutable.

Afin de déterminer si les données sont indépendantes, il est possible d'effectuer un test statistique à cet effet : le two-samples $t$-test.

### 4.2.3 *t*-test de Student

Le $t$-test est un test statistique permettant de déterminer si la statistique d'un test est normalement distribuée. Ici, il sera utilisé pour déterminer si les moyennes des échantillons sont significativement différents.

L'hypothèse nulle est la suivante :

$$H_0 : \mu_1 = \mu_2 \tag{18}$$

Plusieurs versions du test existent avec des hypothèses différentes. L'égalité dans la taille des échantillons considérés ainsi que la quasi-égalité de leur variance (moins de 0.1% de différence) permet alors l'utilisation du test d'indépendance two-samples t-test pour des données de taille et variance égales.

Sous ces hypothèses :

$$T = \frac{\overline{X_1} - \overline{X_2}}{s_p \sqrt{\frac{2}{n}}} \tag{19}$$

avec :

- $\overline{X_i}$ la moyenne de l'échantillon $i$
- n le nombre d'éléments dans l'échantillon
- $s_p^2$ la variance combiné (ou composite), qui est alors l'écart-type estimé depuis les deux populations, et pour des données de même taille, équivaut à la moyenne arithmétique des variances ie
  $$s_p = \sqrt{\frac{s_{X_1}^2 + s_{X_2}^2}{2}}$$
  $s_{X_1}^2$ et $s_{X_2}^2$ sont des estimateurs sans biais de la variance de la population

Le degré de liberté est alors $\nu = 2n - 2$ et correspond au nombre de valeurs dans le calcul final qui peuvent varier sans contrainte.

Il est alors possible de tester l'hypothèse nulle qui sera rejetée avec un degré de confiance de $1 - \alpha$ dans le cas où :

$$|T| \geq t_{1-\alpha/2,\nu}$$

avec $t_{1-\alpha,\nu}$ la valeur critique de la $t$-distribution avec $\nu$ degrés de liberté.

Les p-valeurs sont très élevées, il est alors impossible de rejeter l'hypothèse $H_0$, ce qui conduit à considérer que les données ne sont pas indépendantes.

| $\alpha = 5\%$ | RF Pre | XGB Iter | XGB Expert |
|---|---|---|---|
| Statistique | -0.012 | -0.007 | -0.017 |
| p-value | 0.505 | 0.503 | 0.507 |

TABLE 9 – Statistique et p-values pour le two-samples $t$-test avec $\alpha = 5\%$

Ce critère d'indépendance est malheureusement souvent nécessaire pour tous les tests du même genre. Bien que l'hypothèse ne soit pas vérifiée, il peut quand même être intéressant d'étudier un second test statistique. Si le test de Kolmogorov-Smirnov peut s'apparenter à une norme $\mathscr{L}^1$ entre les deux fonctions de distribution, le critère de Cramér-Von Mises peut lui s'apparenter à une norme $\mathscr{L}^2$.

#### 4.2.4 Critère de Cramér-Von Mises

Le critère de Cramér-Von Mises est comme le test de Kolmogorov-Smirnov, un outil permettant de juger le "goodness-of-fit" (qualité de l'ajustement) d'une fonction de répartition $F^*$ par rapport à une fonction de répartition empirique donnée $F_n$.

Dans le cas où une seule des deux fonctions de répartition n'est empirique et que les observations sont notées $x_1, x_2, ..., x_n$ et classées par ordre croissant :

$$T = \omega_n^2 = \frac{1}{12n} + \sum_{i=1}^{n} \left[ \frac{2i-1}{2n} - F^*(x_i) \right]^2 \tag{20}$$

avec

$$\omega^2 = n \int_{-\infty}^{+\infty} [F_n(x) - F^*(x)]^2 dF^*(x) \tag{21}$$

Dans le cas du test two-samples, les équations sont légèrement différentes.

En considérant $x_1, x_2, ..., x_N$ et $y_1, y_2, ..., y_M$ les deux échantillons ordonnés par ordre croissant et $(r_i)$ et $(s_j)$ les rangs des échantillons de respectivement $x$ et $y$ dans l'ensemble combiné :

$$T = \frac{NM}{N+M}\omega^2 = \frac{U}{NM(N+M)} - \frac{4MN-1}{6(M+N)} \tag{22}$$

avec :

$$U = N \sum_{i=1}^{N} (r_i - i)^2 + M \sum_{j=1}^{M} (s_j - j)^2$$

L'hypothèse est alors rejetée lorsque la valeur de $T$ est trop grande (supérieure au quantile $1-\alpha$ de la distribution de la statistique). Cependant, cette distribution est complexe et ne dispose pas de formule facilement exploitable. Les résultats qui suivent se reposeront sur l'implémentation du critère par le package scipy.

| Loi Beta | Equity | RF Pre | XGB Iter | XGB Expert |
|---|---|---|---|---|
| Statistique | 2.62 | 2.00 | 2.42 | 2.34 |
| p-value | 5.01e-07 | 1.23e-05 | 1.40e-06 | 2.18e-06 |

De manière équivalente au test de Kolmogorov-Smirnov, les p-valeurs sont très faibles et l'hy-

pothèse $H_0$ est fortement rejetée. Il est notable que bien que les p-valeurs soient très faibles, elles sont grandement supérieures à celles obtenues au précédent test, ce qui indique une plus grande tolérance de ce test par rapport au précédent où une déviation, même très ponctuelle, suffisait à pénaliser totalement la distribution testée.

Pour des raisons logicielles, le test two-samples n'a pas pu être effectué mais le résultat aurait probablement été identique à celui du test de Kolmogorov-Smirnov.

Bien qu'un lien avec une distribution connue n'a pas pu être fait, les prédictions des modèles sont incontestablement hautement fidèles aux données empiriques. Dans les cas des modèles testés, le modèle XGB expert se démarque particulièrement en obtenant la meilleure performance par choc sur la quasi-totalité des chocs. Cette domination d'un modèle unique ne sera cependant pas forcément une règle générale au fur et à mesure que de nouveaux modèles et jeux d'hyperparamètres sont testés.

Dans le cas où plusieurs modèles se distinguent en fonction des chocs, un modèle agrégeant les qualités de chaque modèle peut alors être créé, qu'on appelera super-modèle.

**Hypothèse du super-modèle**

Le super-modèle sera obtenu en utilisant le modèle le plus puissant pour un certain type de prédiction. Le critère discriminant choisi a été le choc appliqué et donc pour chaque choc, le super-modèle utilisera le modèle optimal pour ce dit choc.

Soit $X$ des données partitionné par $(X_i)_{i=1,...,K}$ avec $X_i$ ne contenant que des données du choc $i$, alors :

$$SMod(X) = [Max(M_1(X_1), M_2(X_1), ..., M_n(X_1)), ..., Max(M_1(X_K), M_2(X_K), ..., M_n(X_K))] \quad (23)$$

avec $M_i$ le modèle numéro $i$.

Le super-modèle obtenu serait alors théoriquement plus performant que chaque modèle pris individuellement.

Dans le cadre du mémoire, le super-modèle utilisant l'erreur relative par choc comme critère ne ferait que peu de sens. En effet, il serait alors principalement composé du modèle XGB expert et le seul choc qui utiliserait un modèle différent serait le ChocSpreadCDS où la différence en erreur relative avec le XGB expert est inférieure à 1%.

Le critère discriminant peut cependant grandement modifier le ou les modèles choisis pour la construction du super modèle. Les erreurs des modèles étudiés sont relativement proches et donnent pour l'ensemble des résultats très satisfaisants par rapport aux ordres de grandeurs en jeu, une comparaison sur ce seul critère n'est alors pas totalement pertinent. Des critères plus opérationnels peuvent alors être pris en compte, comme la vitesse d'optimisation du modèle, la puissance de calcul nécessaire pour l'entraînement ou encore l'interprétabilité du modèle peuvent être pris en compte.

### 4.3 Agrégation des SCR et erreur finale

Une fois le modèle le plus performant en termes de prédiction choc par choc déterminé, le prochain élément à étudier sont les erreurs liées aux SCR des modules et du BSCR final.

Contrairement au schéma de la figure 2, les chocs appliqués chez CNP Assurances ne vont pas impacter l'intégralité des modules et sous modules de SCR.

La manière dont les SCR modulaires et sous modulaires sont agrégés a déjà été expliquée dans la première partie. L'intérêt se porte alors sur la manière dont les SCR sous modulaires sont calculés.

Les SCR sous modulaires se basent sur la valeur des différents chocs appliqués. Afin de déterminer cette valeur, le $\Delta NAV$ est d'abord calculé et est simplement égal à la différence entre l'Equity pour le choc Central et pour le choc considéré. La valeur du choc est ensuite obtenue en prenant la moyenne de ces $\Delta NAV$ sur l'ensemble des 1000 scénarios, et donc pour un choc $X$ on a :

$$\begin{aligned} \Delta NAV_X &= Equity_{Central} - Equity_X \\ Choc_X &= \overline{\Delta NAV_X} \end{aligned}$$

Le $Choc_X$ sera appelé la valeur du choc $X$ par la suite.

Pour le SCR de marché, les sous modules et chocs concernés sont les suivants :

— $SCR_{Equity}$ : Choc Action Type 1 et Action Type 2
Le SCR est calculé de la manière suivante :

$$SCR_{Equity} = PMP' \tag{24}$$

avec : $P = \left(ChocActionType1, ChocActionType2\right)$
$M = \begin{pmatrix} 1 & 0.75 \\ 0.75 & 1 \end{pmatrix}$

— $SCR_{Spread}$ : Choc Spread CDS, Obligations et Titrisation.
Le SCR est simplement la somme des trois chocs.

— $SCR_{Taux}$ : Choc Hausse Taux et Choc Baisse taux.
Le SCR est égale au plus grand des deux chocs.

— $SCR_{Immo}$

— $SCR_{Change}$

— Le $SCR_{Concentration}$ n'aura pas d'impact sur le cas étudié.

Pour le SCR Vie :

— $SCR_{Rachat}$ : Choc Hausse Rachat, Baisse Rachat et Rachat de masse.
Le SCR est calculé en prenant le maximum des trois chocs.

— $SCR_{Mortalité}$

— $SCR_{Longevité}$

— $SCR_{Frais}$

— $SCR_{VieCat}$

— Les $SCR_{Incapacité}$ et $SCR_{Révision}$ n'impacte pas l'entreprise et ne seront donc pas pris en compte.

Afin d'étudier les SCR sous modulaire, modulaire et BSCR, il est nécessaire de choisir une sensibilité sur lequel les prédictions seront effectuées. Le Closing et la sensibilité hausse taux +50bps seront utilisés afin d'avoir un scénario sans choc particulier et un autre impactant fortement le SCR marché, auquel les assureurs vie sont très sensibles.

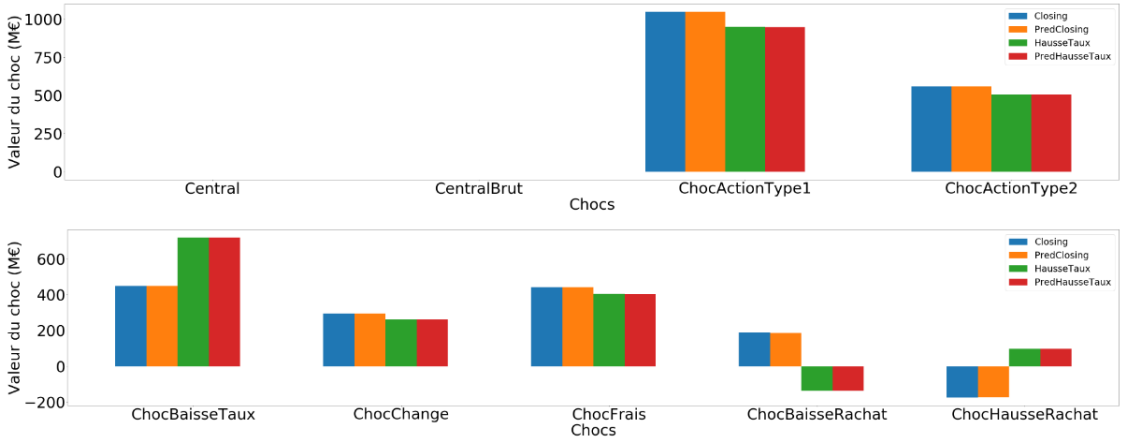



FIGURE 43 – Valeurs des chocs pour le Closing et la sensibilité hausse des taux (part 1)


FIGURE 44 – Valeurs des chocs pour le Closing et la sensibilité hausse des taux (part 2)

Pour chacun des deux chocs, les prédictions sont très proches des valeurs de chocs réelles. Cela confirme alors les résultats obtenus précédemment. L'impact de la hausse des taux est clairement visible, notamment sur les chocs liés aux taux et sur le choc de rachat massif. Avec des prédictions aussi proches des véritables valeurs, les prédictions des SCR sous modulaires sont aussi très proches des vrais SCR sous modulaires. Cela n'est cependant pas le cas pour l'ensemble des sensibilités : en effet, des erreurs assez importantes ont pu être observées aux

figures X et Y. Cela indique alors que le modèle produit des erreurs non négligeables sur certaines sensibilités.

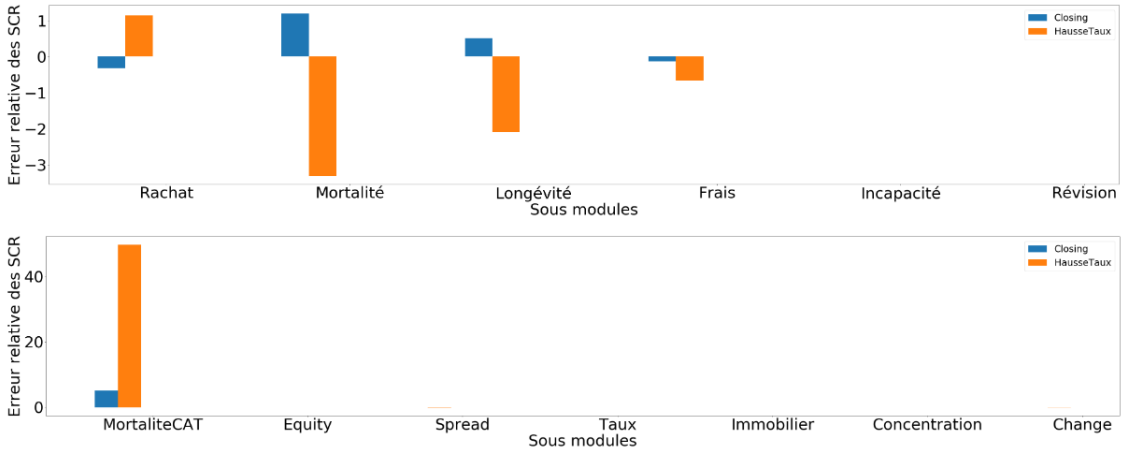



FIGURE 45 – Erreur relative des SCR sous modulaires

Les erreurs relatives sont faibles pour l'ensemble des sous modules, à l'exception du module de mortalité CAT avec une erreur de plus de 40% pour la sensibilité hausse taux. Cependant, le poids de ce module est faible par rapport aux autres modules (moins d'un million), ce qui permet de quand même avoir un modèle performant malgré cette erreur. Pour les SCR du module marché, l'erreur du Mortalité CAT écrase celles des autres mais elles sont aussi très faibles (inférieures à 1%).

Au final, les erreurs sur les modules Vie et Marché ainsi que sur le BSCR sont relativement faibles :

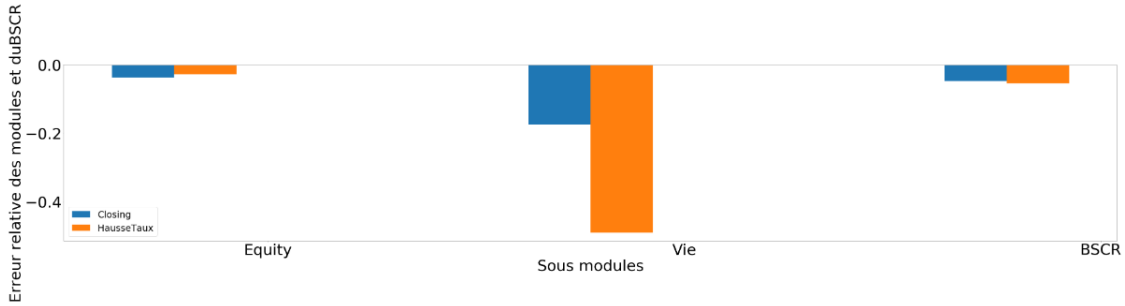



FIGURE 46 – Erreur relative des modules de SCR et du BSCR en M€

Les erreurs sont encore plus faibles que pour les sous modules, les ordres de grandeurs des sous modules les mieux prédits compensant les erreurs des autres et sont alors contenus sous la barre des 1%. Le BSCR ici prédit est une approximation du BSCR, le SCR incorporel n'étant ici pas pris en compte dans son calcul.

Les différents modèles testés ont donc permis une prédiction de l'Equity puis des SCR avec des marges d'erreurs très correctes. Bien que ces performances soient encourageantes, un certain nombre de limites ne permettent pas l'utilisation du modèle tel quel et necessitera des ajustements, parfois majeurs, aux modèles et aux données.

## 4.4 Limites et critiques

Tout au long du mémoire, plusieurs limitations liées aux modèles, à l'infrastructure ou encore aux données ont été relevées. Ces limitations ont empêché d'exploiter complètement la base de données complète de CNP Assurances ainsi que la puissance des modèles utilisés.

### 4.4.1 Limitations physiques

La limitation qui est revenue le plus souvent est celle liée à la puissance de calcul. En effet, la quasi-totalité des calculs ont été lancés sur une plate-forme en ligne de CNP Assurances. Cependant, l'équipe Consolidation de la DRG n'est pas un acteur majeur dans le domaine de la science des données chez CNP Assurances et n'a donc pas accès à une très grande puissance de calcul. De plus, cette puissance doit être partagée entre les différents utilisateurs de la plate-forme de sorte que les travaux de l'un ne soient pas bloquant pour les autres à cause d'une monopolisation des coeurs de la ferme de calcul.

Cette limitation a notamment été contraignante lors de l'optimisation des différents modèles de Machine Learning. Comme évoqué dans la partie sur ces modèles, un entraînement par grid-search sur 600 jeux de paramètres pour le Random Forest menait à des temps de calcul de l'ordre des dizaines d'heures, et ce, lorsque l'optimisation n'était effectuée que sur **1% de la base de données finale**. Lors de l'étude des graphiques liés à l'évolution de l'erreur ou du score en fonction des paramètres, il est clair que des paramètres plus optimaux auraient pu être trouvés si des plages plus étendues et plus fines avaient été utilisées lors de l'optimisation.

En plus de l'optimisation, cette limitation en puissance de calcul s'est aussi fait ressentir lors de l'entraînement des modèles sur la base de données finale, et donc avec cent fois plus lignes de données que par rapport aux optimisations grid-search. Cependant, le temps d'entraînement d'un modèle n'est pas simplement linéairement dépendant au nombre de lignes dans la base de données et les temps nécessaires pour l'entraînement de certains modèles étaient comparables au temps d'optimisation. Les modèles Random Forest avaient par exemple un temps d'entraînement lors de l'optimisation d'au plus deux minutes, cependant, lorsque l'entraînement était effectué sur la base de données complètes, le temps n'était pas de deux cents minutes (trois heures vingt) mais autour de dix à quinze heures. Certains modèles ont même dû être abandonnés à cause d'un temps de calcul excessif. Cela est le cas notamment du Support Vector Regression qui, bien qu'ayant pu être optimisé via grid-search, n'a pas pu être entraîné sur la base réelle (plus d'une semaine de calcul sans résultats).

Une seconde limitation, liée en partie avec la seconde est celle sur l'espace de stockage disponible. Comme évoqué précédemment, la plate-forme utilisée est partagée par de nombreux services et l'espace de stockage disponible est partagée entre les différents utilisateurs. Le chargement des fichiers nécessaire à la création de la base de données finale est très coûteuse en espace de stockage : un seul arrêté avec quarante sensibilités ont nécessité plus de 100Go d'espace de stockage. Sachant que idéalement, l'ensemble des arrêtés disponibles devraient être utilisés afin d'avoir la base de données la plus complète possible, cela reviendrait alors à utiliser **vingt-quatre fois** plus d'espace (quatre arrêtés par an depuis 2016), ce qui n'est simplement pas

concevable avec l'architecture actuelle des données. Des problématiques d'automatisation de l'importation de ces données sont aussi d'actualité, ces données ne pouvant pour le moment que être importées de manière manuelle en les téléchargeant depuis NEMO et en les important dans l'environnement R&D de CNP Assurances.

De plus, cette augmentation massive de la taille de la base de données va d'autant plus accentuer les limitations en puissance de calcul précédent, avec des temps d'optimisation et d'entraînement qui dépasseraient alors certainement les centaines voire les milliers d'heures.

En plus des limites physiques dépendant des caractéristiques de notre environnement, d'autres limitations liées aux modèles ou aux algorithmes utilisés sont aussi présentes.

### 4.4.2 Limitations algorithmiques et liées aux modèles

L'augmentation pure et dure de la puissance de calcul n'est pas un moyen fiable pour régler les problèmes liés à cette limitation. En effet, augmenter la puissance de calcul disponible est très coûteux et ne permet pas toujours de réduire considérablement les temps de calcul. Les améliorations devront donc s'effectuer au niveau algorithmique, soit en utilisant des librairies plus optimisées et en ayant une meilleure compréhension des hyperparamètres des modèles afin de ne pas créer des jeux de paramètres peu pertinents, soit en jouant sur la base de données finale et ne retenant que les variables les plus pertinentes, que ce soit par avis d'expert ou par sélection automatique de variables. Ce sujet sera traité dans la sous partie suivante sur les limitations liées aux choix humains.

L'utilisation d'une base de données incomplète (entre 1% et 20% en fonction de la méthode d'optimisation et du modèle) est très limitant pour l'optimisation des modèles. En effet, certains hyperparamètres des modèles sont dépendants des données utilisées en entrées et les optimiser sur une base réduite mène alors à des paramètres non optimaux pour la base de données finale. C'est le cas notamment du *max_depth*, ou encore du *max_nodes_leaf* qui, pour une base de données plus petite, aura tendance à voir ces paramètres être petits aussi.

Les paramètres obtenus ne seront alors plus forcément adaptés pour la nouvelle base de données, pouvant alors entraîner des cas de sous ou de sur-apprentissage. Une profondeur maximale faible issue d'une petite base de données va empêcher le modèle final d'être suffisamment précis et donc le modèle sera en sous-apprentissage. De manière opposée, la petite taille de la base de données va aussi permettre d'avoir un nombre maximal d'échantillons par feuille faible sans être pénalisant, mais une fois appliqué à la base réelle, une situation de sur-apprentissage pourra émerger. Cette manière de faire ne permet donc pas d'obtenir les véritables paramètres permettant d'optimiser les résultats obtenus sur la base de données complètes. C'est cependant le seul moyen disponible afin de pouvoir quand même optimiser de manière automatique les différents modèles. L'ensemble des paramètres n'étant pas tous sensibles à la taille de la base de données, cet exercice n'est pas inutile et permet d'obtenir une intuition sur certains paramètres.

Pour répondre à cette limitation, la solution idéale serait évidemment d'utiliser la base de données complète. Dans le cadre du mémoire, des jeux de paramètres par avis d'expert ont été

déduits des optimisations par grid-search et itératif pour le XGBoost afin de proposer des paramètres plus pertinents pour les données réelles. Bien qu'elle ait permis d'obtenir un modèle très performant, cette solution nécessite une forte implication de l'utilisateur et requiert alors ce dernier d'avoir une expertise non négligeable dans le domaine, ce qui n'est pas forcément le cas de la population visée par cet outil.

### 4.4.3 Limitations liées aux processus et données utilisées

Lors de la création de la base de données, plusieurs choix ont été effectués pouvant être discutable :

— Choix des fichiers/variables retenues (non transformées)

— Choix de la transformation des variables

Lors de l'analyse des fichiers d'entrée, des choix ont été effectués pour décider quelles variables ou indicateurs allaient être retenus pour la création de la base de données. Différents choix ont donc dû être effectués, que ce soit pour les données du model point d'actif ou des ESG mais par précaution il était possible de retenir l'ensemble des variables (sauf celles spécifiques au processus de CNP Assurances et n'apportant alors aucune information). Cependant, pour les fichiers IndicateursAvecPB, le choix était beaucoup moins évident. En effet, les fichiers IndicateurAvecPB sont des **sorties**(secondaires) du modèle ALM de CNP Assurances, qui peuvent être utilisés pour calculer de manière **déterministe** l'Equity du choc considéré. Il peut alors sembler étrange d'utiliser une sortie, pour ensuite prédire un élément de cette même dite sortie. Cependant, en utilisant uniquement les entrées du modèle ALM, il aurait été impossible d'obtenir un résultat avec des erreurs acceptables. En effet, durant les calculs ALM, de nombreuses projections, utilisant des tables d'hypothèses variées sont en jeu, et les relations sont si complexes que les modèles n'auraient pas pu en tirer une structure sous-jacente, surtout compte-tenu des limites précédentes sur la puissance de calcul et la taille de la base de données.

Afin de contourner ce problème, des variables ont été retenues afin d'aider les modèles à prédire les bons résultats. Il n'était cependant pas possible de laisser trop de variables sous leur forme VAN étant donné qu'une simple régression linéaire réussirait à deviner le calcul de l'Equity et obtenir des résultats quasiment parfaits (moins de 10 euros d'écart entre la prédiction et la valeur réelle). Il a donc été nécessaire d'effectuer une transformation sur ces variables de sorties afin de les rendre à la fois plus interprétable par l'utilisateur, et moins informatifs pour le modèle, et le choix a donc été de les transformer sous forme de taux pour la période considérée (par exemple, le montant de rachat a été transformé en taux de rachat pour la période du T4 2021).

En plus de cette transformation en taux, de nombreuses transformations et choix ont été faits sur les variables provenant du model point d'actif et des ESG. Ces choix ont été faits par avis d'expert lorsqu'il s'agissait de regrouper des variables ou des modalités, de les rendre proportionnelles à une autre mais d'autres choix auraient très pu mener à des résultats plus pertinents, plus interprétable ou alors moins denses et donc plus faciles à entraîner. Pour certaines autres variables, un ensemble de statistiques ont été choisies pour les représenter mais au-

cune étude n'a été effectuée pour le choix de ces dernières et le choix a été porté sur des statistiques communes, facilement interprétable et ne nécessitant pas une étude approfondie des variables.

Cette liste de limitations et de critiques n'est bien entendu pas exhaustive, mais permet d'émettre des réserves claires sur les résultats obtenus.

En plus des critiques sur les limites que présente ce mémoire, des ouvertures et pistes d'exploration existent aussi.

## 4.5 Pistes d'explorations potentielles

### 4.5.1 Autres modèles de machine learning et deep learning

Les modèles utilisés sont parmi les plus populaires en machine learning, notamment pour des problèmes de régression. D'autres modèles de machine learning existent bien évidemment, notamment le Support Vector Regression cité précédemment, ou des modèles de régression linéaire pénalisés tels que l'ElasticNet. En plus des autres modèles classiques utilisables, la catégorie des modèles dits de deep learning, qui sont des algorithmes essayant d'imiter le comportement des cerveaux humains via l'utilisation de neurones artificiels avec par exemple les modèles de réseaux de neurones, peuvent aussi obtenir des résultats pertinents. Cependant, les modèles de deep learning se reposent sur une logique différente de ceux des modèles de machine learning classique. Avec ces derniers, l'expert va extraire les features (ou variables) qu'il voudra utiliser pour entraîner son modèle comme cela a pu être fait dans la partie création de la base de données du mémoire. Les limites liées à cette extraction manuelle ont été énoncées précédemment, et tant qu'un humain sera en charge de faire les différents choix liés à cet exercice, ces limites seront toujours présentes.

Dans le cas du deep learning, le modèle va extraire de la donnée brute les informations qu'il juge lui même pertinente. Cela va alors permettre au modèle de ne récupérer uniquement les informations utiles, se débarrassant alors des données n'apportant que du bruit et se créant alors ses propres variables pour une prédiction optimale. Dans le cadre d'une reconnaissance d'images, cela revient alors à donner l'image directement au modèle sans en vouloir en extraire des données au préalable. Dans le cas de la prédiction de SCR, les fichiers d'entrée seront donnés avec des modifications minimales au modèle. Cependant, les données utilisées en entrées sont déjà sous forme de variables et peuvent alors être assimilées à une extraction préalable par un utilisateur humain. De plus, certaines variables trop informatives (les variables d'IndicateurAvecPB) ne peuvent pas être utilisées telles quelles et doivent alors être quand même transformées.

Parmi les modèles de deep learning, les modèles de classification sont parmi les plus étudiés et les plus performants, notamment dans la reconnaissance d'images. Il est alors possible de dériver ces modèles de classification pour un usage plus axé sur la régression.

### 4.5.2 Utilisation de modèles de machine learning pour la classification

Durant ce mémoire, l'ensemble des modèles utilisés ont été des modèles de régression. Ce choix est le plus évident quand il s'agit de prédire une valeur continue et permet d'avoir un modèle potentiellement parfait. Cependant, ces modèles de régression ont aussi leurs limites : en utilisant des régressions linéaires, ils sont alors soumis au risque de sur-apprentissage et sont sensibles au bruit des données.

Le principe d'utiliser des modèles de classification pour un problème de régression est assez simple. Au lieu de vouloir prédire une valeur précise, une prédiction d'appartenance à un intervalle sera effectuée. Dans le contexte d'une estimation de SCR, une prédiction très précise n'est pas forcément pertinente. En effet, cet outil a été créé dans le but d'avoir une estimation rapide de l'impact d'un choc ou d'un changement non prévu. Ces informations sont souvent alors communiquées sous forme de variations en point de ratio de solvabilité. Il est alors nécessaire de seulement estimer l'impact à un certain niveau de précision ce qui permet de ne considérer qu'un nombre réduit de catégories dans le cas d'une classification. Pour obtenir une précision au pourcent près pour un ordre de grandeur donné, il suffirait alors d'avoir au plus cent catégories pour obtenir des informations sur une déviation jusqu'à plus ou moins 50% à cette maille. Le nombre de catégories considérées peut être davantage réduit si l'on ne considère que certains paliers de déviation, par exemple : 0-1%, 1-5%, 5-10%, 10-20% et 20% et plus. En considérant aussi les déviations négatives, on obtiendrait alors seulement 10 catégories ce qui devient alors raisonnable pour entraîner les modèles de classification multi-classes.

# 5 Conclusion générale

En utilisant les données officielles de CNP Assurances, une base de données qui a servi de base d'entraînement et de validation pour les différents modèles de machine learning a pu être construite. Cette base a été formée en utilisant les données du model point d'actifs, du marché ainsi que les résultats ALM de l'entreprise. Afin de pouvoir utiliser ces ressources, des analyses et des transformations ont été effectuées, à la fois pour corriger les différentes erreurs qu'elles auraient pu produire si utilisées brutes, mais aussi pour leur donner une forme facilement utilisable et compréhensible pour les modèles de machine learning.

Les deux grands modèles testés ont été les modèles de type Random Forest et ceux de type eXtreme Gradient Boost, qui ont été ensuite combinés avec des techniques de réduction de variables dont le Recursive Feature Elimination et la régression LASSO. Les différents modèles ont été optimisés en utilisant le MAE comme métrique, avec comme but de prédire l'Equity (la VIF) des données, et donc en déduire les SCR sous modulaires, modulaires et BSCR. Le modèle obtenant finalement le meilleur score fut le modèle XGBoost utilisant les hyperparamètres choisis par avis d'expert.

Ce modèle n'est cependant pas celui ayant été le plus performant sur l'ensemble des chocs, avec un modèle Random Forest lui étant légèrement supérieur pour le choc SpreadCDS. Dans le cas étudié, la création d'un super-modèle n'a pas été jugée pertinente étant donné la faible amélioration que celle ci engendrerait mais reste une piste à explorer dans le cas où un futur modèle viendrait concurrencer celui en place de manière plus conséquente.

Le modèle ainsi obtenu permet de prédire le BSCR avec une erreur relative inférieure à 1%, performance qui devrait être largement suffisante dans le cadre d'une estimation ponctuelle de SCR pour lequel cet outil a été créé. Cependant, ce résultat et outil ne sont pas totalement utilisables comme tels, étant donné que cette performance est notamment due à de nombreuses hypothèses très fortes dont les limites ont été décrites dans la partie précédente. De plus, l'outil tel qu'il est n'est pas aisément manipulable par un utilisateur, notamment à cause du relativement grand nombre de variables explicatives nécessaires ainsi que de leurs interdépendances (faire varier une seule variable reviendrait à ignorer les dépendances qui peuvent être très fortes avec les autres, ce qui peut aboutir à des résultats éloignés de la réalité).

Les résultats obtenus sont donc très encourageants mais nécessite un approfondissement certain avant de pouvoir être utilisé en parallèle des outils de production. Cependant, cet outil permet de créer une brique de base sur laquelle de futurs projets pourront se construire, et confirme la pertinence d'une telle méthode comme alternative au processus actuel.

# Références


[1] Tarek AOUDI (2022), *Optimisation du ratio de solvabilité, pour les contrats d'épargne, en contexte de taux bas*, Mémoire d'actuaire, ENSAE Paris

[2] Yannig Goude, *Méthodes d'ensemble et forêts aléatoires*

[3] Simon COUZI (2021), *Rapport de stage M2 CNP Assurances*

[4] Cosma Shalizi (2006), *Statistics 36-350, Carnegie Mellon University*

[5] Rui Guo et al (2020) *Degradation state recognition of piston pump based on ICEEMDAN and XGBoost*

[6] Eric Leroux(2009), *De Solvabilité I à Solvabilité II*

[7] Will Kenton (2022), *Goodness of fit, Investopedia*

[8] Eric Feigelson and G. Jogesh Babu, *Kolmogorov-Smirnov Test, Center for Astrostatistics, Penn State University*

[9] ACPR (2011), *QIS 5*

[10] ACPR (2015), *Notice Solvabilité II : Calcul SCR, ACPR*

[11] Aymric Kamega (2015), *Introduction à Solvabilité 2, Applications de mesure des risques*

[12] Cédrin Morin (2015),*Le reporting multinormes en assurance vie - Enjeux et éléments de convergence des référentiels Solvabilité 2 et IFRS 4 phase 2*, Mémoire d'actuaire, ISUP

[13] Hugo Cochard (2022), *Utilisation du machine learning dans la détermination d'une allocation optimale sous Solvabilité II*, Mémoire d'actuaire, ISFA

[14] Emilie Soix (2018), *Estimation du ratio de solvabilité à l'aide de méthodes d'apprentissage statistique supervisé*, Mémoire d'actuaire, ISFA

[15] Isabelle Guyon et al (2002), *Gene Selection for Cancer Classification using Support Vector Machines*

[16] Institut des Actuaire (2016), *Le risque opérationnel, un nouveau challenge pour l'actuaire*

[17] Laurent Devineau (2017), *Modélisation et Agrégation des risques*

[18] Kla Kouadio (2018), *Méthodes Prospectives de Calcul de SCRs et Applications*, Mémoire d'actuaire, ISUP

[19] Institute and Faculty of Actuaries (2016) *Solvency II - Life Insurance*

[20] Dossier technique Optimind (2007), *Solvabilité II et les modèles internes*

[21] Mariette Awad Rahul Khanna (2015), *Support Vector Regression, Efficient Learning Machines*

[22] T.W. Anderson (1962), *On the Distribution of the Two-Sample Cramer-von Mises Criterion*

[23] L. Baringhausa N. Henzeb (2016), *Cramer–von Mises distance: Probabilistic interpretation, confidence intervals, and neighborhood-of-model validation*

[24] Stacey Ronaghan (2018), *Random Forest and feature importance*

[25] M. G. Kendall (1938) *A new measure of rank correlation*

[26] Pearson (1895), *Note on Regression and Inheritance in the Case of Two Parents*

[27] Gregory Gutin, Anders Yeo, Alexey Zverovich (2002), *Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP*

[28] Robert Tibshirani (1996), *Regression Shrinkage and Selection via the Lasso*

[29] Gregory W. Corder, Dale I. Foreman (2014), *Nonparametric Statistics: A Step-by-Step Approach*

[30] Smirnov (1948), *Table for Estimating the Goodness of Fit of Empirical Distributions*

[31] M. Stone (1974), *Cross-Validatory Choice and Assessment of Statistical Predictions*