DAUPHINE UNIVERSITÉ PARIS

Master Actuariat de Dauphine

DAUPHINE UNIVERSITÉ PARIS

Mémoire présenté devant l'Université Paris Dauphine pour l'obtention du diplôme du Master Actuariat et l'admission à l'Institut des Actuaires

le 26/01/2018

Par : Junqi WEI

Titre: Homogénéité des risques: application à la sinistralité en assurance non-vie

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

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

Membre présent du jury de l'Institut des Actuaires :

Signature :

Entreprise :

Nom : PwC France

Signature :

Directeur de mémoire en entreprise :

Nom : Alexandre VEBER

Signature :

Membres présents du jury du Master Actuariat de Dauphine :

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

Signature du responsable entreprise :

Secrétariat :

Bibliothèque :

Signature du candidat :

Remerciements

J'adresse mes sincères remerciements à Alexandre VEBER qui a accepté d'encadrer ce mémoire, pour ses encouragements, ses conseils avisés.

Je remercie aussi Emmanuel DUBREUIL qui m'a offert cette opportunité de réaliser ce mémoire, pour son encouragement et sa confiance.

Je tiens à remercier Aurélien PACARD pour son implication, sa disponibilité et ses conseils.

J'adresse également ma reconnaissance à Angelina ROCHE, ma tutrice académique pour sa patience et le temps qu'elle a consacré à corriger mes fautes d'orthographe.

J'adresse aussi mes remerciements à Patrice BERTRAND pour son aide précieuse.

Je remercie tout particulièrement Guillaume LAUTREY et Léa LAUTREY pour leur collaboration et soutien tout au long de cette aventure.

Je pense aussi à Paul OTTOU, Laurent DEBRIL, Angéla PANGO, Vincent NOEL et Alexandre GRAS pour leur soutien et encouragement.

J'adresse ma gratitude à l'ensemble des équipes du service RVMS de PwC France, pour leur accueil chaleureux, leur disponibilité et leur entier soutien.

Pour finir, je remercie ma famille et mes amis pour leur soutien et encouragement.

Trop nombreux sont ceux que je n'ai pu nommer, qu'ils trouvent ici l'expression de ma gratitude.

Résumé

En assurance non-vie, la modélisation de la prime pure repose sur des hypothèses fortes sur la distribution des risques. L'hétérogénéité des risques peut potentiellement remettre en question ces hypothèses et ainsi la robustesse du modèle sous-jacent.

C'est dans ce cadre que s'inscrit ce mémoire, dont l'objectif consiste à détecter les groupes de risques homogènes dans un but de mieux prédire les sinistres d'une manière automatique.

A cet effet, deux principales approches sont présentées : l'approche *a posteriori* et l'approche *a priori*. En s'appuyant sur les algorithmes d'apprentissage automatique, différentes méthodes dans les deux approches sont proposées.

Elles sont mises en place sur un portefeuille d'assurance automobile. Les résultats ainsi obtenus sont discutés. Nous en déduisons que la détection des groupes de risques homogènes permet effectivement d'augmenter les capacités prédictives des modèles.

Mots-clés :

Tarification non-vie, segmentation, homogénéité des risques, algorithmes d'apprentissage automatique, CART, XGBoost, analyse en composantes principales, réseau de neurones artificiels, auto-encodeur, K-means, la carte de Kohonen, modèle de mélange gaussien.

Abstract

In non-life insurance, the rating models of the pure premium are based on the hypotheses of the distribution of the risks. The heterogeneity of the risks may challenge these hypotheses and therefore the robustness of the rating models considered.

This paper is written to examine this issue. Its objective consists of detecting homogeneous risk groups with the aim of a better prediction of the losses in an automatic way.

Two primary approaches - the *a priori* approach and the *a posteriori* approach. Several machine learning based methods are proposed.

They are implemented on a motor insurance portfolio. The results thus obtained are discussed. It is concluded that the detection of homogeneous risk groups contribute to improving the robustness of the rating models and thus to achieving better predictive capacities.

Key words :

Non-life rating, segmentation, homogeneity of risks, machine learning, CART, XGBoost, principal component analysis (PCA), artificial neural networks (ANNs) , auto-encoder, K-means, self-organizing map (SOM), Gaussian mixture model(GMM).

Synthèse

Problématiques

Ce mémoire est consacré à la détection de groupes de risques homogènes. En assurance non-vie, la modélisation de la prime pure repose sur des hypothèses fortes sur la distribution des risques. D'une manière générale, la prime pure est déterminée par le produit de l'espérance conditionnelle du nombre de sinistres et l'espérance conditionnelle du montant des sinistres individuels. Afin d'estimer le nombre de sinistres, les actuaires utilisent souvent des méthodes basées sur les modèles linéaires généralisés en supposant que la distribution du nombre de sinistres suit une loi de Poisson, ou une loi binomiale négative, etc. Quant à l'estimation du montants des sinistres, les mêmes méthodes sont implémentées mais en supposant que la distribution des montants individuels suit une loi Gamma ou log-normale [1], etc.

L'hétérogénéité des risques peut potentiellement remettre en question les hypothèses sur la distribution des risques et ainsi la robustesse du modèle sous-jacent. En effet, si les hypothèses sur la distribution des risques est fausse, alors le modèle ainsi construit risque de ne pas être robuste et d'être biaisé [2]. Par conséquent, afin de garantir la robustesse d'un modèle, il nécessite une hypothèse pertinente sur la distribution des risques qui colle aux données réelles. Toutefois il n'est pas toujours évident d'avoir une bonne hypothèse sur la distribution des risques, car justement l'hétérogénéité des risques peut conduire à une distribution très complexe. L'allure de la distribution du montant des sinistres n'a toujours pas de forme simple et il est peu pertinent d'ajuster une seule loi de distribution.

Divide et impera [3]. Comme le dit Philippe II de Macédoine, il est naturel

1. Il est à noter que comme la loi log-normale ne fait pas partie de la famille exponentielle, une transformation est nécessaire pour les modèles linéaires généralisés.

2. Gilles Dupin, Alain Monfort, Jean-Pierre Verle, *Robust inference in rating models*, Proceedings of the 34th ASTIN Colloquium, 2003.

3. Diviser pour mieux régner.

de penser qu'en partitionnant l'ensemble du portefeuille en groupes de risques homogènes, les risques au sein de chaque groupe seront identiquement distribués ou presque. L'objective de ce mémoire consiste alors à détecter les groupes de risques homogènes d'une manière automatique, de sorte qu'au sein de chaque groupe les risques soient identiquement distribués. Ainsi, avec les hypothèses plus adéquates sur la distribution, les modèles ainsi construits seront plus robustes, ce qui permettra d'obtenir de meilleures prédictions.

Les apports des algorithmes d'apprentissage automatique

Les méthodes basées sur les algorithmes d'apprentissage automatique, pourront servir à résoudre nos problématiques. En particulier, les algorithmes de classification non-supervisées comme les K-means et SOM peuvent être implémentés pour identifier les groupes de risques homogènes. Le CART permet de faire une régression, à l'issue duquel les observations au sein d'une même feuille peuvent aussi être vues comme un groupe homogène. L'algorithme supervisé XGBoost peut jouer le rôle des modèles linéaires généralisés.

En s'appuyant sur les algorithmes d'apprentissage automatique, deux approches principales et diverses méthodes sont proposées.

L'approche *a posteriori*

La première est l'approche *a posteriori*. Dans laquelle, nous travaillons principalement sur les résidus de déviance. L'expression "*a posteriori*" provient du fait que dans un premier temps un modèle de régression est calibré sur l'ensemble des données ; une fois ledit modèle construit, les résidus sont obtenus ; dans un second temps, les groupes homogènes sont identifiés grâce aux classifications effectuées sur les résidus. Une régression est ensuite effectuée respectivement sur chaque groupe pour prédire les sinistres via l'algorithme d'XGBoost. Deux méthodes sont proposées dans cette approche. Elles sont

- GMM sur les résidus de déviance ;
- K-means sur les résidus de déviance.

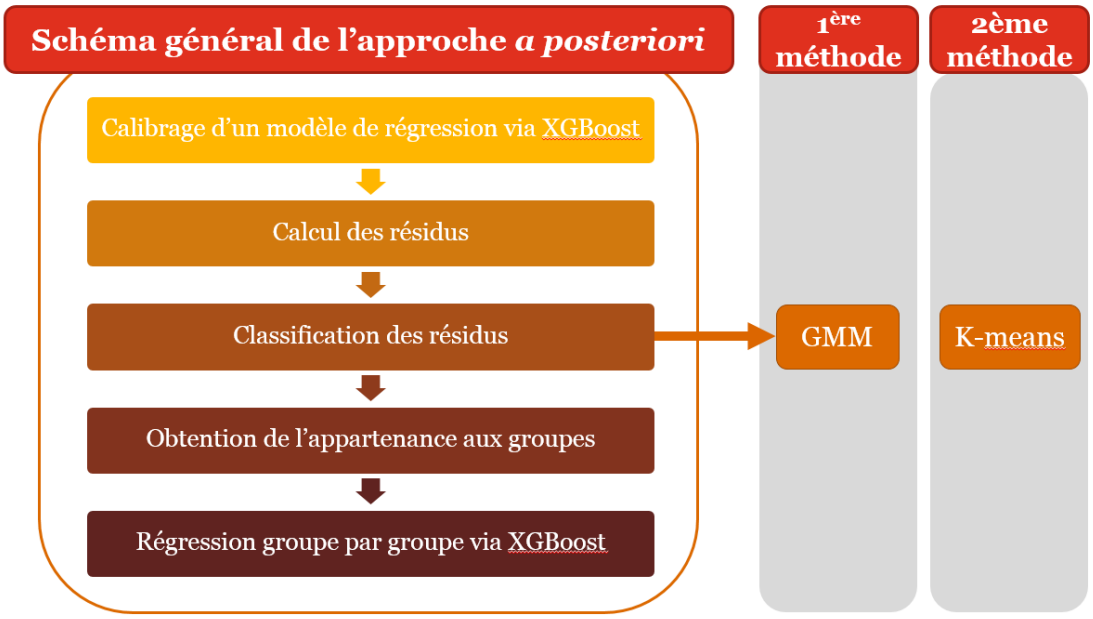

FIGURE 1 – Schéma général de l'approche *a posteriori* et les méthodes associées

L'approche *a priori*

La deuxième approche correspond à l'approche *a priori*. D'une manière générale, elle se compose de deux étapes :

- Dans un premier temps, une classification est effectuée selon les caractéristiques des assurés ou celles des sinistres, à l'issue de laquelle des groupes de risques homogènes sont obtenus ;
- Dans un second temps, une régression est effectuée pour prédire les sinistres via l'algorithme d'XGBoost respectivement sur chaque groupe obtenu par l'étape précédente.

Plusieurs méthodes sont proposées dans l'approche *a priori*. Elles sont

- ACP – K-means ;
- Auto-encodeur – K-means ;
- ACP – SOM ;
- ACP – SOM – K-means ;
- CART sur les résidus obtenus par le modèle naïf [4] ;

4. Nous avons inclus cette méthode dans l'approche *a priori* au lieu de l'approche *a posteriori* car il n'y a pas eu lieu un calibrage du modèle à proprement parler puisque les prédictions sont toutes égales à la moyenne des montants des sinistres dans l'échantillon d'apprentissage.

- CART pour prédire les montants des sinistres ;
- GMM sur log(montants des sinistres) ;
- GMM sur les résidus obtenus par le modèle naïf.

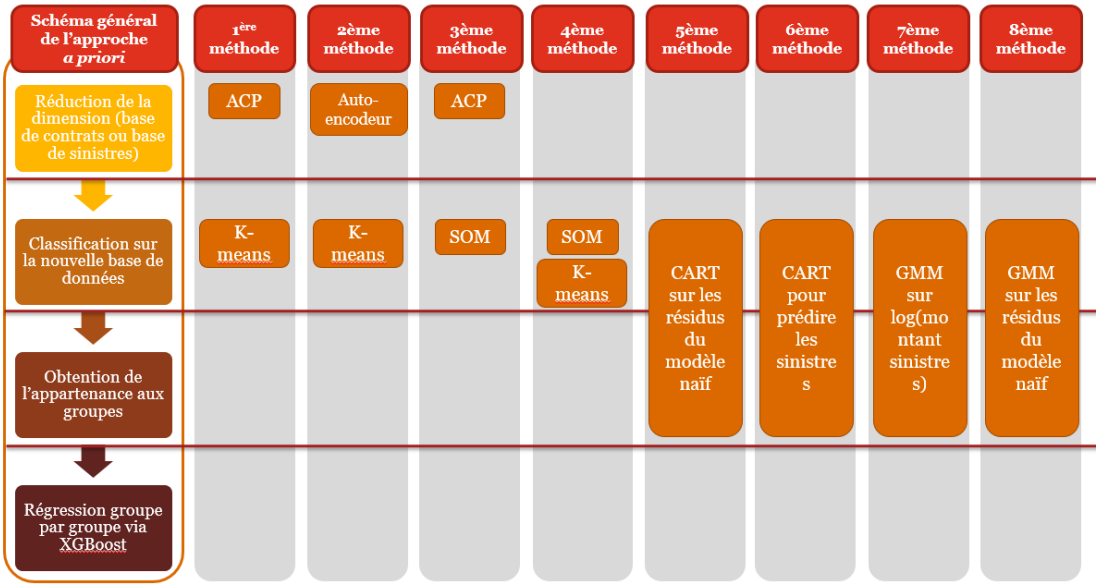

FIGURE 2 – Schéma général de l'approche *a priori* et les méthodes associées

Résultats

Les résultats obtenus sont prometteurs et correspondent à nos attentes. Une segmentation en groupes de risques homogènes s'est avérée être propice à mieux prédire la sinistralité. Nous avons obtenu une diminution significative de la déviance, MSE et MAE grâce à la segmentation, que ce soit via l'approche *a posteriori*, ou via l'approche *a priori*. Parmi toutes les méthodes proposées dans les deux approches, globalement la méthode "GMM sur log(montant des sinistres)" dans l'approche *a priori* réalise les meilleures performances, ce qui nous invite à porter notre choix pour cette méthode.

Summary

Issue

This paper is devoted to the detection of homogeneous risk groups. In non-life insurance, the rating models of the pure premium are based on the strong assumptions on the distribution of the risks. In general, the pure premium is determined by the product of the conditional expectation of the number of claims and the conditional expectation of the amount of individual claim. In order to estimate the number of claims, actuaries use the methods based on the generalized linear models (GLM) on the assumption that the number of claims has a Poisson distribution, or a Negative Binomial distribution, etc. As for the estimation of the amount of the individual claim, the same methods are implemented yet on the assumption that the individual claim is Gamma-distributed or log-normal distributed [5], etc.

The heterogeneity of the risks may challenge the assumptions on the distribution of the risks and therefore the robustness of the underlying model. In fact, if the assumptions on the distribution of the risks are mistaken, then the build model may not be robust and be biased [6]. Therefore, in order to guarantee the robustness of a rating model, it requires an appropriate assumption on the distribution of the risks which matches well the actual data. However, it is not always obvious to have a well-suited assumption on the distribution of the risks, because the heterogeneity of the risks may lead to a very complex distribution. The look of the distribution of the amount of claims does not always have a simple form and it would hardly be appropriate to fit a mere distribution.

Divide et impera [7]. Just as Philippe II de Macédoine put it, it would be rea-

5. Note that since the log-normal distribution does not belong to the exponential family, a transformation is needed to perform the generalized linear models (GLM).

6. Gilles Dupin, Alain Monfort, Jean-Pierre Verle, *Robust inference in rating models*, Proceedings of the 34th ASTIN Colloquium, 2003.

7. Divide and rule.

sonable to expect that by partitioning the entire portfolio into homogeneous risk groups, the risks inside of each group will be identically distributed or almost. The objective of this paper consist in detecting the homogeneous risk groups in an automatic way, such that inside of each group the risks are identically distributed. Thus, with more appropriate assumptions on the distribution, the underlying models will be more robust, thus achieving better predictions.

With the help of Machine learning

The machine learning-based methods may contribute to resolving our problems. In particular, the algorithms of unsupervised classification such as K-means and SOM can be implemented in order to identify the homogeneous risk groups. The CART can be used to do a regression, as a result of which the observations inside of a same leave can be considered as a homogeneous risk group. The supervised algorithm XGBoost can play the role of the generalized linear models (GLM).

With the help of machine leaning, two main approaches and a variety of methods are proposed.

The *a posteriori* approach

The first approach consists of the *a posteriori* approach. In this approach, we focus on the deviance residuals. The expression "*a posteriori*" is due to the fact that as a first step, a regression model is fitted on the entire portfolio ; once the above-mentioned model is constructed, the residuals are calculated ; as a second step, the homogeneous risk groups are identified thanks to the classifications performed on the residuals. A regression model is then fitted on each group respectively to predict the claims via XGBoost. Two methods are proposed in this approach. They are

- GMM on the deviance residuals ;
- K-means on the deviance residuals.

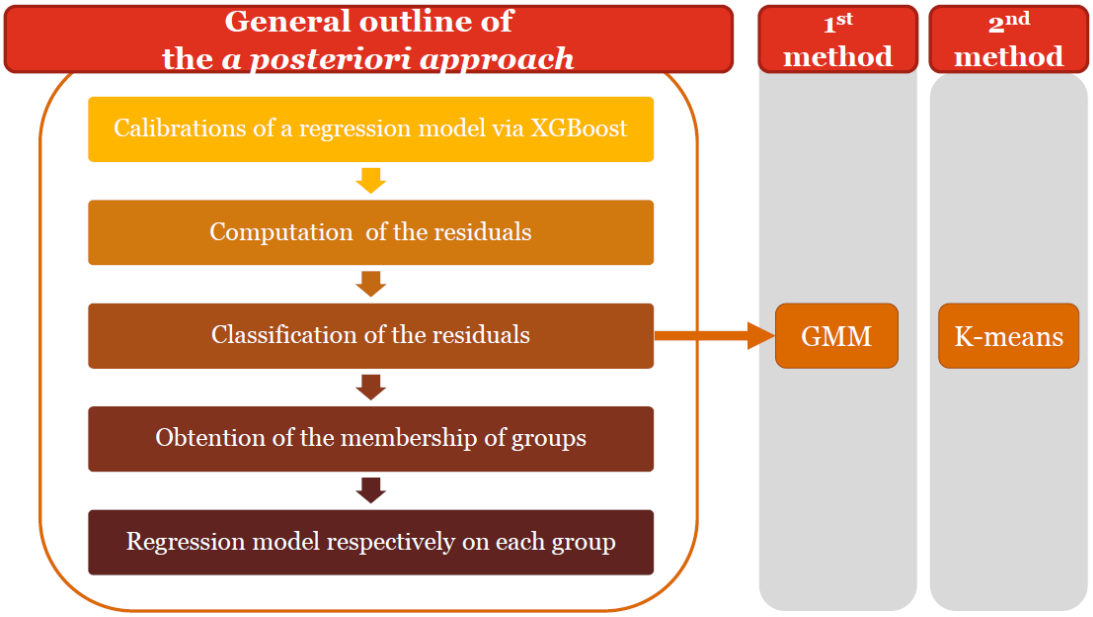

FIGURE 3 – General outline of the *a posteriori* approach and its associated methods

The *a priori* approach

The second approach consists of the *a priori* approach. In general, it is composed of two steps :

- As a first step, a classification is performed according to the characteristics of the policyholders or those of the claims, as a result of which, the homogeneous risk groups are obtained ;
- As a second step, a regression model is fitted to predict the claims via XGBoost respectively on each group obtained by the previous step.

A variety of methods are proposed in this approach. They are

- PCA – K-means ;
- Auto-encoder – K-means ;
- PCA – SOM ;
- PCA – SOM – K-means ;
- CART on the residuals obtained by the naïve model [8] ;

8. This method is included in the prior approche instead of in the posterior approach is due to the fact that strictly speaking we haven't fit a model since the predictions are equal to the mean of the amount of the claims in the training set.

- CART to predict the claim amount ;
- GMM on log(claim amount) ;
- GMM on the residuals obtained by the naïve model.

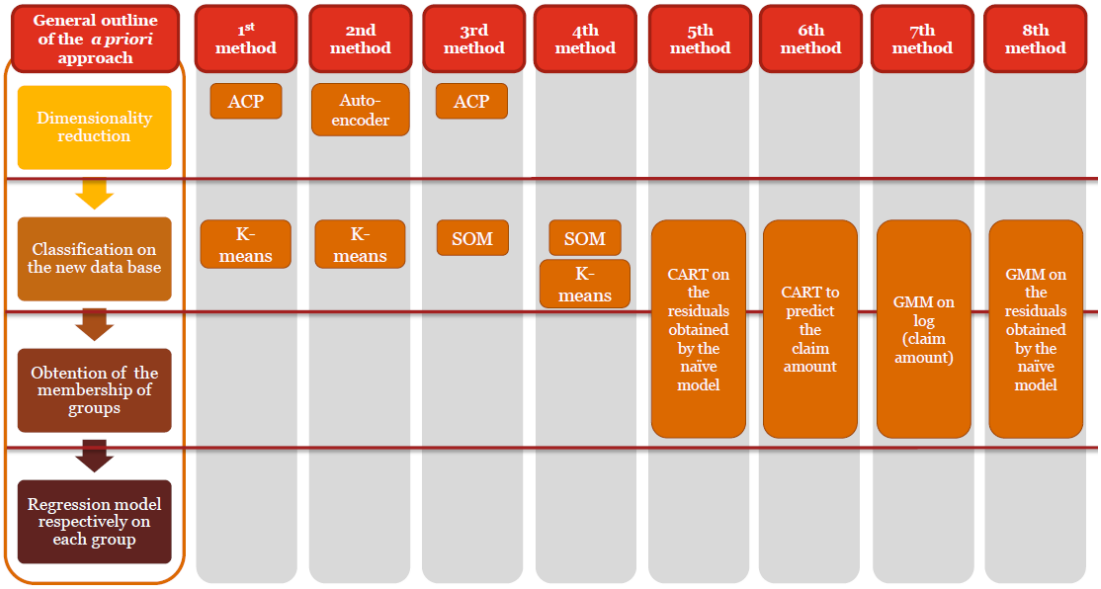

FIGURE 4 – General outline of the *a priori* approach and its associated methods

Résultats

The results are promising and are aligned with our expectations. A segmentation en homogeneous risk groups is proved to be helpful to better predict the losses. We have achieved a significant decrease in the deviance, MSE and MAE thanks to the segmentation, whether via the *a posteriori* approach or via the *a priori* approach. Among all the methods proposed in these two approaches, the method "GMM on log(amount of claims)" in the *a priori* approach achieve the best performances overall, which invites us to opt for it.

Table des matières

Chapitre 1

Introduction générale

En tarification non-vie, la modélisation de la prime pure repose sur des hypothèses fortes sur la distribution des risques. La prime pure est souvent déterminée par le produit de l'espérance conditionnelle du nombre de sinistres et l'espérance conditionnelle du montant des sinistres individuels. Afin d'estimer le nombre de sinistres, les actuaires utilisent souvent les méthodes basées sur les modèles linéaires généralisés en supposant que la distribution du nombre de sinistres suit une loi de Poisson, ou une loi binomiale négative, ou encore une autre loi adéquate. Quant à l'estimation du montant des sinistres, les mêmes méthodes sont implémentées mais en supposant que la distribution des montants individuels suit une loi Gamma ou log-normale [1], etc.

L'hétérogénéité des risques peut potentiellement remettre en question les hypothèses sur la distribution des risques et ainsi la robustesse du modèle sous-jacent. En effet, si les hypothèses sur la distribution des risques est fausse, alors le modèle ainsi construit risque de ne pas être robuste et d'être biaisé [2]. Par conséquent, afin de garantir la robustesse d'un modèle, il nécessite une hypothèse pertinente sur la distribution des risques qui colle aux données réelles. Toutefois il n'est pas toujours évident d'avoir une bonne hypothèse sur la distribution des risques, car justement l'hétérogénéité des risques peut conduire à une distribution très complexe. L'allure de la distribution du montant des sinistres n'a toujours pas de forme simple et il est peu pertinent d'ajuster une seule loi de distribution.

1. Il est à noter que comme la loi log-normale ne fait pas partie de la famille exponentielle, une transformation est nécessaire pour les modèles linéaires généralisés.

2. Gilles Dupin, Alain Monfort, Jean-Pierre Verle, *Robust inference in rating models*, Proceedings of the 34th ASTIN Colloquium, 2003.

Divide et impera[3]. Comme le dit Philippe II de Macédoine, il est naturel de penser qu'en partitionnant l'ensemble du portefeuille en groupes de risques homogènes, les risques au sein de chaque groupe seront identiquement distribués. L'objective de ce mémoire consiste alors à détecter les groupes de risques homogènes d'une manière automatique, de sorte qu'au sein de chaque groupe les risques soient identiquement distribués. Ainsi, avec les hypothèses plus adéquates sur la distribution, les modèles ainsi construits seront plus robustes, ce qui permettra d'obtenir de meilleures prédictions.

Les méthodes basées sur les algorithmes d'apprentissage automatique, pourront servir à résoudre nos problématiques. En particulier, les algorithmes de classification non-supervisés comme les K-means et SOM peuvent être implémentés pour identifier les groupes de risques homogènes. L'algorithme supervisé XGBoost peut jouer le rôle des modèles linéaires généralisés.

En s'appuyant sur les algorithmes d'apprentissage automatique, deux approches principales et diverses méthodes sont proposées.

La première est l'approche *a posteriori*. Dans laquelle, nous travaillons principalement sur les résidus de déviance. L'expression "*a posteriori*" provient du fait que dans un premier temps un modèle de régression est calibré sur l'ensemble des données ; une fois ledit modèle construit, les résidus sont obtenus. Dans un second temps, les groupes homogènes sont identifiés grâce aux classifications effectuées sur les résidus. Une régression est ensuite effectuée respectivement sur chaque groupe pour prédire les sinistres via l'algorithme d'XGBoost.

La deuxième approche correspond à l'approche *a priori*. D'une manière générale, elle se compose de deux étapes :

- Dans un premier temps, une classification est effectuée selon les caractéristiques des assurés ou celles des sinistres, à l'issue de laquelle des groupes de risques homogènes sont obtenus ;
- Dans un second temps, une régression est effectuée pour prédire les sinistres via l'algorithme d'XGBoost respectivement sur chaque groupe obtenu par l'étape précédente.

Le mémoire s'articulera ainsi en 4 parties :

- Au chapitre 2, nous remettons la problématique dans son contexte puis définirons la méthodologie de travail ;

3. Diviser pour mieux régner.

- Les théories et divers algorithmes d'apprentissage automatique sont ensuite présentés au chapitre 3 ;
- Puis au chapitre 4, nous mettrons en application les deux approches principales et différentes méthodes ;
- Nous concluons ce mémoire au chapitre 5 en indiquant les limites des méthodes proposées et en envisageant quelques axes d'amélioration.

Chapitre 2

Contexte général et motivation

L'objectif de ce chapitre est d'introduire la problématique liée à l'hétérogénéité des risques et de définir la méthodologie. Dans la première section de ce chapitre, nous donnerons une introduction du calcul de prime pure en tarification non-vie et nous nous attacherons à revoir les hypothèses sous-jacentes qui soulèveront la problématique de l'hétérogénéité des risques. L'objectif de ce mémoire est ensuite défini dans la deuxième section. Nous terminons ce chapitre sur la présentation de la méthodologie adoptée.

2.1 Problématique

2.1.1 Les principes généraux de calcul de prime pure en tarification non-vie

D'une manière générale, la prime pure en tarification non-vie correspond à l'espérance mathématique du coût des sinistres déclarés à l'assureur durant une période considérée [1].

Supposons que l'assureur possède un portefeuille composé d'un grand nombre de polices. Soit S_i, $i = 1, \ldots, n$, le montant total de sinistres payé par l'assureur pour la police numéro i durant une période considérée. Désignons par $\bar{S}$ le coût

1. Michel Denuit, Arthur Charpentier, *Mathématiques de l'assurance non-vie, Tome II : Tarification et provisionnement*, 2009.

moyen des sinistres par police :

$$\bar{S} = \frac{1}{n} \sum_{i=1}^{n} S_i$$

Si de plus les S_i sont indépendants, identiquement distribués et de variance finie, alors la loi des grands nombres nous dit que

$$\bar{S} = \frac{1}{n} \sum_{i=1}^{n} S_i \xrightarrow{n \to \infty} \mathbb{E}[S_1] \quad \text{en probabilité.}$$

Cette propriété s'interprète comme suivant : lorsque le portefeuille est assez grand le coût moyen de sinistres par police converge en probabilité vers l'espérance mathématique du montant total de sinistres de la i-ème police. Ceci justifie la détermination de la prime pure (*pure premium*) par l'espérance mathématique de S_1 :

$$\text{La prime pure} = \mathbb{E}[S_1].$$

Le modèle collectif et lois composées

Concentrons-nous à présent sur le calcul de l'espérance mathématique de S_i. D'après les hypothèses ci-dessus, comme les S_i sont indépendants et identiquement distribués, S_i est désigné par S dorénavant. Soit N le nombre de sinistres qui ont eu lieu durant la période considérée et soit Y_j le montant du j-ème sinistre. Alors le coût pour l'assureur durant cette période vaut

$$S = \begin{cases} \sum_{j=1}^{N} Y_j & \text{si } N \neq 0 \\ 0 & \text{si } N = 0 \end{cases}$$

Remarquons que les N et Y_j sont des variables aléatoires. Nous parlons alors de "lois composées" car le coût de l'assureur se compose d'une loi de comptage pour modéliser le nombre de sinistres, et d'une autre loi souvent continue pour modéliser les montants de sinistres.

De plus, si les hypothèses suivantes sont satisfaites :

1. Les montants de sinistres Y_j sont indépendants ;
2. Les montants de sinistres Y_j sont identiquement distribués ;
3. N et Y_j sont indépendants.

Alors nous pouvons montrer que

$$\begin{aligned} \text{La prime pure} &= \mathbb{E}[S] \\ &= \mathbb{E}[N] \cdot \mathbb{E}[Y_j] \end{aligned}$$

En d'autres termes, la prime pure correspond au produit de l'espérance du nombre de sinistres ($\mathbb{E}[N]$) et l'espérance du montant de sinistre individuel ($\mathbb{E}[Y_j]$). Pour estimer l'espérance du nombre de sinistres, les actuaires calibrent un modèle de fréquence. Quant à l'estimation du montant des sinistres individuels, un modèle de coût moyen est calibré.

2.1.2 Remise en question des hypothèses sous-jacentes

Examinons de près les hypothèses sous-jacentes, nous pouvons faire les remarques suivantes :

- Tout d'abord, afin d'appliquer la loi des grands nombres, il faut que la taille du portefeuille soit suffisamment grande.
- Deuxièmement, les montants des sinistres, i.e. les risques dans le portefeuille, doivent être indépendants. Cette hypothèse peut ne pas être vérifiée. Par exemple, un assureur possède un portefeuille d'assurances multirisques habitation. Une grande partie de ce portefeuille est composée de immeubles situés dans le même quartier. En cas de déluge, ces risques ne sont plus indépendants. Il en est de même pour les catastrophes naturelles comme les tremblements de terre et les inondations. Nous n'approfondissons pas cette problématique de la dépendance des risques car ce n'est pas le sujet de ce mémoire.
- Troisièmement, les montants des sinistres, i.e. les risques au sein du portefeuille doivent être identiquement distribués, ce qui n'est pas toujours le cas. Cela s'explique par deux facteurs principaux :
  - **l'hétérogénéité des profils de risque** : au sein d'un même portefeuille, certains individus ont plus tendance à avoir des accidents alors que d'autres ont moins tendance. Il n'est pas difficile d'imaginer que Madame Prudente et Monsieur Étourdi n'auront pas les mêmes types de sinistres.

— **l'hétérogénéité des risques couverts** : Cette hétérogénéité est dû aux caractéristiques diverses et variées des risques couverts. Par exemple, il est évident que les risques de "vol" et ceux de "catastrophe naturelle" n'auront pas la même distribution.

La première - l'hétérogénéité des profils de risque peut aussi avoir une influence sur la dernière - l'hétérogénéité des risques couverts.

Du fait de l'hétérogénéité, les risques ne sont donc pas identiquement distribués. Les hypothèses actuarielles sur lesquelles s'appuient la modélisation de la prime pure peuvent potentiellement être remises en question. D'un côté, la distribution non-identique des risques implique la variation de l'espérance des sinistres (les sinistres n'auront pas la même espérance) ; D'autre côté, cela signifie que les sinistres sont de variance différentes. Nous parlons alors de l'hétéroscédasticité. Une collection de variables aléatoires est hétéroscédastique, s'il y a des sous-populations qui ont des variabilités différentes des autres [2].

Pour y remédier, les actuaires ont recours à l'espérance conditionnelle. Les méthodes classiques de tarification comme l'utilisation des méthodes basées sur les modèles linéaires généralisés (GLM) sont ainsi développées. Toutefois, les modèles linéaires généralisés ne garantissent pas la résolution de la problématique de l'hétéroscédasticité. C'est dans ce contexte que s'écrit ce mémoire : identifier les groupes de risques homoscédastiques de sorte que les risques au sein de chaque groupe soient identiquement distribués.

2.1.3 Hétérogénéité des risques : un exemple introductif

Donnons un exemple concret. En assurance automobile, la garantie "bris de glace" prendra en charge des frais de remplacement ou de réparation d'une vitrée cassée sur une voiture. Cette garantie couvre un champ très vaste. Il peut s'agir d'un petit trou sur le pare-brise. Auquel cas une simple réparation suffit. Le pare-brise peut aussi être brisé entièrement et il nécessite alors un remplacement complet. Ces deux cas présentent des caractéristiques très différentes : une réparation de pare-brise a une fréquence élevée mais un coût faible comparé à un remplacement de pare-brise qui a une fréquence faible mais un coût élevé. Comme les sinistres associés ont des distributions distinctes, il est alors peu pertinent de les modéliser de la même manière.

2. https ://fr.wikipedia.org/wiki/Hétéroscédasticité

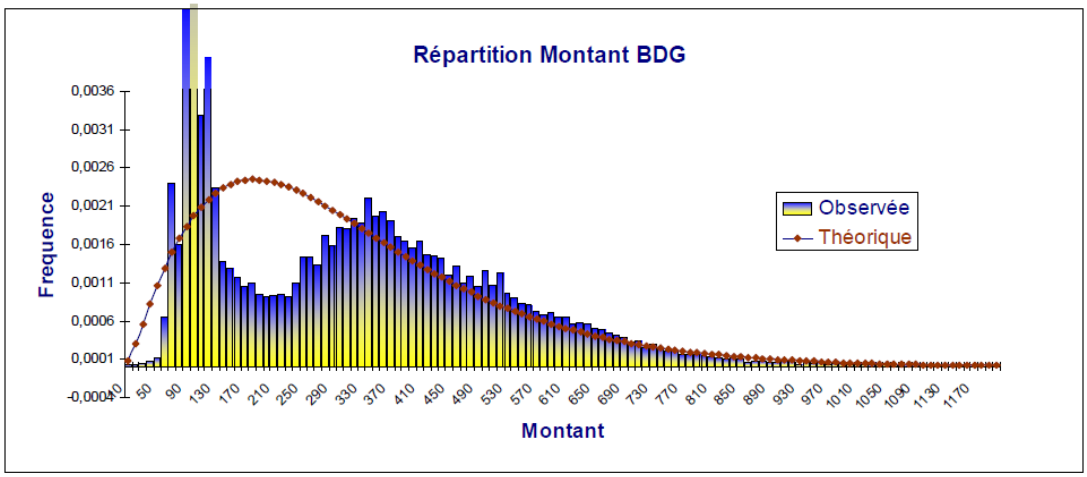

FIGURE 2.1 – Histogramme des montants de bris de glace

La Figure 2.1 [3] nous montre l'histogramme des montants de bris de glace d'un portefeuille d'un assureur. Nous distinguons deux pics correspondant respectivement aux montants de réparation et ceux de remplacement. Il est alors souhaitable de les séparer en deux parties et de calibrer un modèle de régression respectivement pour la réparation et pour le remplacement comme l'illustre la Figure 2.2.

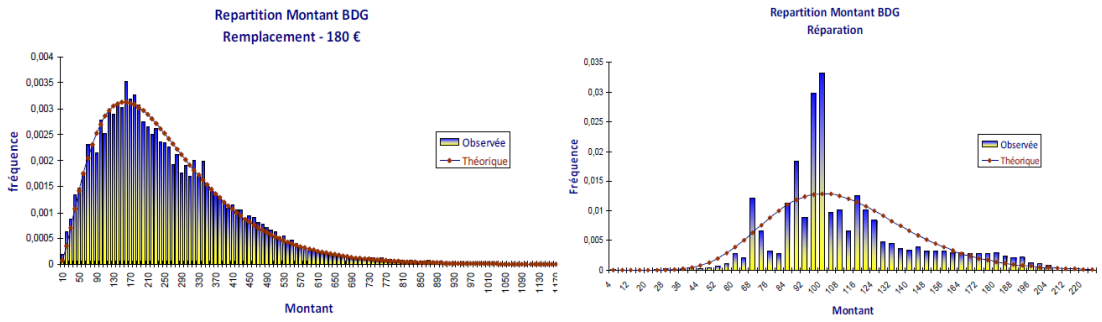

FIGURE 2.2 – Histogrammes des montants de bris de glace en remplacement et en réparation

Toutefois, en pratique, il n'est pas toujours évident de parvenir à une telle séparation. Dans cet exemple, *a priori*, nous connaissons la nette distinction entre les risques liés à la réparation et ceux liés au remplacement. Toutefois, dans de

3. Source : MAITI Maher Maxim, *Modélisation de la prime pure en assurance automobile*, 2012

nombreux autres cas, la distinction n'est pas aussi claire. Par exemple, en assurance construction, les actuaires ne sont pas forcément capables de distinguer les différents risques liés à la responsabilité civile décennale (RCD).

2.1.4 Hétérogénéité des risques et modèle de mélange de lois

Mathématiquement, l'hétérogénéité des risques peut se formaliser par le modèle de mélange de lois.

Définition Soit Θ une variable aléatoire décrivant l'hétérogénéité du portefeuille. Supposons que conditionnellement à $\Theta = \theta$, la loi de la variable aléatoire X se caractérise par la fonction de répartition $F(.|\theta)$:

$$F(x|\theta) = \mathbb{P}(X \leq x|\Theta = \theta)$$

La loi de la variable aléatoire X est alors caractérisée par la fonction de répartition suivante :

$$\mathbb{E}[\mathbb{P}(X \leq x|\Theta)] = \int_{\theta \in \mathbb{R}} F(x|\theta)\, \mathrm{d}F_{\Theta}(\theta)$$

où F_{Θ} désigne la fonction de répartition de Θ.

La formule ci-dessus s'interprète comme une moyenne pondérée des fonctions de répartition conditionnelles.

Reprenons l'exemple précédent à la sous-section 2.1.3. La distribution du nombre de sinistres est un mélange de deux lois de Poisson différentes. [4]. Si nous calibrons un modèle de fréquence par une seule loi de Poisson, cela revient à faire une hypothèse sous-jacente sur l'homogénéité des risques, ce qui est une approximation un peu forte qui conduira à des biais dans le modèle. Il en est de même pour la distribution des montants de sinistres.

4. En tarification non-vie, le nombre de sinistres est souvent modélisé par un modèle de Poisson.

2.2 Objectif

Comme nous avons vu dans les sections précédentes, l'hétérogénéité des risques peut poser un grand nombre de problèmes. En particulier, les hypothèses sous-jacentes sur la distribution des risques peuvent potentiellement être mises en cause. La modélisation de la prime pure ou l'estimation des sinistralités sera biaisée. Face à un portefeuille mélangeant des risques à caractéristiques diverses et variées, il est donc nécessaire de le partitionner en groupes de risques homogènes.

Le terme "homogénéité" peut recouvrer différentes significations. Il peut s'agit de :

- **l'homogénéité de nature.** C'est-à-dire qu'au sein d'un groupe, les risques sont supposés être de la même nature. Par exemple, le "risque vol" et le "risque catastrophe naturelle" ne peuvent pas être classés dans le même groupe car ils ne sont pas de même nature et ils possèdent des caractéristiques très différentes.
- **l'homogénéité du point de vue statistique.** C'est-à-dire que les risques au sein d'un même groupe sont identiquement distribués. Dans l'exemple mentionné à la sous-section 2.1.3, les risques liés au "bris de glace" sont répartis en deux groupes : les risques liés au "réparation" et ceux liés au "remplacement". Les risques liés au "réparation" ont à peu près la même distribution de sinistralité. Il en est de même pour les risques liés au "remplacement".

Dans ce mémoire, nous essayerons d'identifier les groupes tout en respectant l'homogénéité de nature et l'homogénéité du point de vue statistique. Il convient cependant de remarquer qu'à l'issue de la segmentation, les risques au sein d'un même groupe ne seront pas tous ressemblants en tout point, et il restera un certain degré d'hétérogénéité. En effet, l'objectif de ce mémoire consiste à détecter les groupes de risques homogènes d'une manière automatique, de sorte qu'au sein de chaque groupe les risques soient identiquement distribués. Ainsi, avec les hypothèses plus adéquates sur la distribution, les modèles ainsi construits seront plus robustes, ce qui permet de mieux prédire la sinistralité grâce à cette segmentation, c'est-à-dire que l'erreur globale de modèles de régression modélisés sur chaque groupe soit optimale.

Mesures d'évaluation

Étant donnée l'objectif évoqué ci-dessus, nous allons choisir la ventilation optimale en comparant les erreurs de régression obtenues par différents modèles que nous détaillerons plus tard. Trois mesures d'évaluation sont considérées afin d'avoir une vision plus fiable des performances de nos modèles. Elles sont :

- **La déviance Gamma**. Nous comparons le modèle construit avec le modèle de référence qui est le "modèle saturé". Il possède autant de paramètres que d'observations et permet de prédire parfaitement les sinistres, c'est-à-dire que toutes les prédictions sont égales aux vraies valeurs de la variable à prédire. La fonction de vraisemblance de ce modèle se note $\mathcal{L}(y|y)$ et celle du modèle calibré par les actuaires se note $\mathcal{L}(\hat{y}|y)$. Il est évident que le modèle construit prédira moins bien que le modèle saturé. Toutefois, il est souhaitable que la vraisemblance du modèle construit soit aussi proche de celle du modèle saturé, i.e. $\mathcal{L}(\hat{y}|y) \approx \mathcal{L}(y|y)$. Nous pouvons le quantifier par la statistique du rapport de vraisemblance

$$\Gamma = \frac{\mathcal{L}(y|y)}{\mathcal{L}(\hat{y}|y)}$$

En modélisant les deux termes en logarithme, nous obtenons la formule équivalente suivante

$$\begin{aligned} \log(\Gamma) &= \log(\frac{\mathcal{L}(y|y)}{\mathcal{L}(\hat{y}|y)}) \\ &= \log(\mathcal{L}(y|y)) - \log(\mathcal{L}(\hat{y}|y)) \end{aligned}$$

Si le modèle est bien calibré, nous nous attendons à ce que Γ, ou $\log(\Gamma)$ ne soit pas très grande. Ainsi nous définissons la déviance $D = 2\log(\Gamma)$. Les actuaires utilisent souvent cette quantité pour mesurer la pertinence du modèle calibré. Plus petite est la déviance, plus adéquat est le modèle.

Nous pouvons montrer que la déviance Gamma vaut

$$2\sum_{i=1}^{N}\left(-\log(\frac{y_i}{\hat{y}_i}) + \frac{y_i - \hat{y}_i}{\hat{y}_i}\right)$$

où y_i désigne la vraie valeur de la variable à prédire pour l'observation i, $\hat{y}_i$ sa prédiction, et N le nombre total d'observations.

Il convient de remarquer que l'analyse de la déviance ne se limite pas aux modèles linéaires généralisés. Tant qu'il y a une hypothèse sur la distribution, nous pouvons calculer la déviance [5].

En actuariat tarification non-vie, les actuaires utilisent souvent la loi Gamma pour calibrer les montants des sinistres dans le cadre des modèles linéaires généralisés. D'autres lois comme log-normale sont aussi envisageables [6], mais moins fréquemment utilisées que la loi Gamma. Ainsi, nous avons opté pour la déviance Gamma comme l'une des mesures d'évaluation cohérentes tout au long de ce mémoire. Certes, c'est une hypothèse assez forte, la loi Gamma reste la distribution la plus utilisée comparée à toutes les autres lois. De plus, pour avoir une "seconde opinion", deux autres mesures sont ajoutées pour remédier à l'inadéquation éventuelle de cette hypothèse.

- **L'erreur quadratique moyenne, ou mean squared error (MSE)**. Elle est l'une des mesures les plus souvent utilisées pour caractériser la qualité d'un modèle de régression. Elle se définit comme

$$\mathbf{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2.$$

Il convient de remarquer qu'elle est équivalente à une déviance gaussienne.

- **L'erreur absolue moyenne, ou mean absolute error (MAE).** L'autre mesure que nous rencontrons souvent pour caractériser la qualité d'un modèle de régression est l'erreur absolue moyenne. Elle se définit comme

$$\mathbf{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|.$$

5. Dan Tevet, *And the winner is... ? How to pick a better model : Part 2 Goodness-of-fit and internal stability*, Verisk Insurance Solutions.

6. Il est à noter que comme la loi log-normale ne fait pas partie de la famille exponentielle, une transformation est nécessaire pour les modèles linéaires généralisés.

2.3 Méthodologie

En pratique, après le calibrage d'un modèle de régression, il est nécessaire de faire une analyse des résidus, qui permet de valider le modèle ou comprendre éventuellement les raisons pour lesquelles le modèle est mal ajusté.

Résidus de déviance

Il y a plusieurs définitions différentes de résidus. L'un des résidus évoqués souvent par les actuaires est le "résidu de déviance". Comme son nom l'indique, les résidus de déviance font référence à la déviance mentionnée à la section précédente. La déviance D peut se décomposer en une somme de d_i. Les d_i peuvent être considérés comme la contribution de chaque observation y_i à la déviance totale :

$$D = \sum_{i=1}^{N} d_i$$

Les résidus de déviance sont définis alors comme suivante :

$$r_i = \text{signe}(y_i - \hat{y}_i)\sqrt{d_i}$$

En particulier, nous pouvons montrer que les résidus de déviance de Gamma vaut

$$r_i = \text{signe}(y_i - \hat{y}_i)\sqrt{|\log(\frac{y_i}{\hat{y}_i}) - \frac{y_i - \hat{y}_i}{\hat{y}_i}|}.$$

L'approche *a posteriori*

Lorsque l'analyse des résidus montre des biais, il est probable que l'hypothèse de loi ou celle sur la distribution identique des risques ne soit pas vérifiée. Il est alors possible d'analyser les résidus pour en déduire des groupes de loi identique. Des modèles de régression sont ensuite calibrés respectivement sur chaque groupe. Une fois que ces modèles sont construits, nous pouvons refaire une analyse des résidus et nous obtenons de nouveaux groupes. Des nouveaux modèles sont ensuite calibrés sur ces nouveaux groupes obtenus, et ainsi de suite. La Figure 2.3 illustre un tel processus.

FIGURE 2.3 – L'approche *a posteriori* (nous supposons que chaque fois nous obtenons 2 sous-groupes)

Nous parlons alors de l'approche *a posteriori* car dans un premier temps nous calibrons un modèle sur l'ensemble des données de contrats ; une fois ledit modèle construit, nous obtenons des résidus de ce modèle ; dans un second temps, nous identifions les groupes en effectuant des classifications sur ces résidus ; puis nous ré-calibrons des modèles par groupe obtenu, ainsi de suite. Il s'agit d'un algorithme récursif et très coûteux.

De l'approche *a posteriori* à l'approche *a priori*

L'approche *a posteriori* décrite ci-dessus peut être longue et laborieuse, ce qui conduit à penser à l'approche *a priori* : au lieu d'itérer entre calibrages des modèles et identifications de nouveaux groupes, ce serait plus judicieux et beaucoup moins

coûteux d'identifier les groupes de risques homogènes dès le départ, avant même le calibrage du modèle.

Pour cela, nous pouvons explorer les informations disponibles dans la base de sinistres ou dans la base de contrats en s'appuyant principalement sur les algorithmes de classification non supervisées. Dans l'idéal, il parait plus simple de détecter les groupes de risques homogènes en travaillant avec la base de sinistres. Toutefois, il existe également un lien de causalité plus ou moins fort entre les caractéristiques des assurés et les risques couverts. Il est naturel d'imaginer que les individus ayant des caractéristiques similaires auront des sinistralités relativement proches. Donc les mêmes algorithmes peuvent être appliqués sur la base de contrats pour atteindre le même but.

Détaillons à présent les différentes étapes de modélisation dans l'approche *a priori*.

Description des étapes de modélisation

En général, la modélisation se compose de deux étapes principales :

- Dans un premier temps, une classification est effectuée sur les observations dans la base de données [7]. A l'issue de cette classification, nous obtenons des groupes homogènes ;
- Dans un second temps, une régression est effectuée pour prédire les montants des sinistres sur la base de contrats via l'algorithme d'XGBoost respectivement sur chaque groupe relativement homogène, obtenu par l'étape précédente.

Nous avons choisi l'XGBoost comme notre modèle de régression au lieu des modèles linéaires généralisés (GLM) principalement pour les raisons suivantes :

- L'XGBoost est beaucoup plus simple à appliquer et aussi beaucoup plus rapide que les méthodes d'estimation basées sur des modèles linéaires généralisés. D'une manière générale, les modèles linéaires généralisés sont assez lourds à appliquer. Par exemple, avant d'ajuster les modèles linéaires généralisés, une étape préalable de sélection de variables est indispensable, mais elle est laborieuse et pesante ; alors que l'XGBoost permet de sélectionner les variables pertinentes automatiquement ;
- L'XGBoost, avec le paramètre de la profondeur de l'arbre égale à 1, est proche d'un modèle linéaire généralisé sans les termes d'interation. En effet,

7. Il peut s'agit d'une base de contrats ou d'une base de sinistres.

en tarification, compte tenu du nombre de variables et de leurs différentes modalités ou valeurs, il est impossible de tester les significativités de tous les termes d'interation. Prenons un exemple : avec 10 variables explicatives, chacune possédant 5 modalités, il est alors nécessaire de tester $5^{10} = 9\,765\,625$ d'interations. Par conséquent, dans un modèle linéaire généralisé, les termes d'interation, lorsqu'il y en a, sont souvent donnés par les avis d'expert ou spécifiées *a priori* par les actuaires [8]. Ainsi nous choisissons de négliger les termes d'interation ;

- De plus, notre objectif est d'identifier les groupes de risques homogènes puis calibrer des modèles de régression respectivement sur chaque groupe. Nous espérons que les prédictions des sinistres ainsi obtenues sur-performent celles obtenues en calibrant un seul modèle de régression sur l'ensemble de données. Donc le sujet central de cette étude est d'identifier les groupes homogènes, et non le choix du modèle de régression. L'algorithme d'XGBoost forme alors notre cadre cohérent de test de la pertinence de partitionnement de groupes tout au long de ce mémoire.

Pour les raisons citées ci-dessus, nous avons opté pour l'algorithme d'XGBoost comme notre modèle de régression. Les formalisations mathématiques de l'XGBoost seront présentées au Chapitre 2.

Une étape préliminaire : transformer les variables qualitatives en variables quantitatives

En pratique, dans la base de données, nous avons les variables quantitatives et qualitatives à la fois. Or la plupart des algorithmes que nous avons besoin d'appliquer ne prennent que des valeurs numériques comme entrée. Par conséquent, en plus des deux étapes principales évoquées précédemment, une étape de préparation des données est préalablement nécessaire pour transformer les données.

Il parait évident qu'il est peu pertinent de convertir directement les modalités de variables qualitatives en valeur numérique comme $1, 2, 3, 4\ldots$ car les modalités ne sont pas nécessairement comparables ni ordonnées. Par exemple, pour les couleurs de voitures, en remplaçant la couleur bleu par 1, blanc par 2, rouge par 3, nous sous-entendons qu'il existe un ordre entre les couleurs, et blanc est plus proche de bleu que rouge, ce qui est absurde.

8. Antoine Paglia, Martial V.PHELIPPE-GUINVARC'H, *Tarification des risques en assurance non-vie, une approche par modèle d'apprentissage statistique*, Bulletin Français d'Actuariat, vol. 11, numéro 22, juillet - décembre 2011, pp. 49 - 81.

En général, nous pouvons recourir à trois méthodes :

- **Encodage one-hot**, ou "one-hot encoding" en anglais. Concrètement, étant donné une variable qualitative, nous générons une colonne booléenne pour chaque modalité. Prenons un exemple simple. Nous disposons d'un échantillon dont la variable "couleur" a trois modalités : bleu, blanc et rouge.

Numéro de l'échantillon	Couleur
1	Bleu
2	Rouge
3	Blanc
4	Blanc
5	Bleu
6	Bleu
7	Rouge

FIGURE 2.4 – L'exemple de la variable "couleur" ayant trois modalités

L'encodage one-hot va créer un tableau binaire suivant :

Numéro de l'échantillon	Bleu	Blanc	Rouge
1	1	0	0
2	0	0	1
3	0	1	0
4	0	1	0
5	1	0	0
6	1	0	0
7	0	0	1

FIGURE 2.5 – Illustration de l'encodage one-hot

L'encodage one-hot est une transformation assez populaire à la présence des variables qualitatives. Toutefois, il augmente la dimension de données. Dans l'exemple ci-dessus, la dimension de données augmente de 1 à 3 suite à la transformation de l'encodage one-hot. Lorsque les variables qualitatives dans la base de données possèdent un nombre élevé de modalités, la transformation via l'encodage one-hot conduira une augmentation significative de dimension. Cela n'est pas souhaitable surtout lorsque la base de données d'origine est déjà de grande dimension. C'est pour cette raison-là que nous avons décidé de ne pas retenir cette méthode dans la suite du mémoire.

- **Analyse des correspondances multiples (ACM).** L'analyse des correspondances multiples fait partie de la famille de l'analyse en composantes principales. L'ACM s'applique lorsque la base de données est décrite uniquement par des variables qualitatives. Or en pratique, la base de données est souvent composée de variables qualitatives et quantitatives à la fois. Afin d'appliquer l'ACM, il faut d'abord discrétiser les variables quantitatives et les mettre en classes, ce qui fait perdre beaucoup d'informations ; ensuite l'encodage one-hot mentionné ci-dessus est effectué pour obtenir un tableau disjonctif complet ou un tableau de Burt ; puis nous mettons en œuvre l'ACM sur le tableau de Burt ; enfin nous conservons les k premiers composantes principales selon certains critères prédéfinis (par exemple, préserver 90% de variance). La base de données obtenue suite à l'ACM sera en numérique.

  Cette méthode a été testée sur notre base de données. La base de données résultant de l'ACM reste en très grande dimension tout en conservant seulement 50% de variance. Par conséquent, cette méthode sera aussi exclue de ce mémoire.

- **L'espérance de la variable à prédire.** Étant donné une variable quantitative et l'une de ses modalités, nous remplaçons cette dernière par l'espérance de la variable à prédire par rapport à la modalité. Prenons le même exemple que l'encodage one-hot. La variable à prédire se note y.

Numéro de l'échantillon	Couleur	y
1	Bleu	150
2	Rouge	120
3	Blanc	140
4	Blanc	100
5	Bleu	160
6	Bleu	180
7	Rouge	110

FIGURE 2.6

En remplaçant chaque modalité par l'espérance de la variable à prédire par rapport à cette modalité, nous obtenons :

Numéro de l'échantillon	Couleur	y
1	Bleu	150
5	Bleu	160
6	Bleu	180

$\mathbf{E}[y \mid Bleu] = \frac{150 + 160 + 180}{3} = 163,3$

Numéro de l'échantillon	Couleur	y
2	Rouge	120
7	Rouge	110

$\mathbf{E}[y \mid Rouge] = \frac{120 + 110}{2} = 115$

Numéro de l'échantillon	Couleur	y
3	Blanc	140
4	Blanc	100

$\mathbf{E}[y \mid Blanc] = \frac{140 + 100}{2} = 120$

FIGURE 2.7 – Illustration de la méthode "l'espérance de la variable à prédire"

Enfin nous obtenons le tableau transformé :

Numéro de l'échantillon	Couleur	y
1	163,3	150
2	115	120
3	120	130
4	120	100
5	163,3	160
6	163,3	180
7	115	110

FIGURE 2.8 – Illustration de la méthode "l'espérance de la variable à prédire"

La mise en œuvre de cette méthode est très répandue dans de nombreux concours de *data-science* notamment les compétitions de *Kaggle*. De plus, cette méthode est facile à appliquer. D'une part, nous ne perdons pas d'information ; d'autre part, nous n'augmentons pas la dimension des données. Ainsi dans la suite de ce mémoire, nous choisissons cette méthode pour transformer les variables qualitatives en numériques.

Encore une autre étape préliminaire : réduction de la dimension

Après l'étape de transformation des données en numériques, une autre étape de réduction de la dimension est également indispensable, car le nombre important des variables explicatives peut rendre difficile l'obtention de regroupements pertinents, notamment dû au "fléau de la grande dimension", ou "*curse of dimensionality*" en anglais.Deux différentes méthodes seront appliquées : l'analyse en composantes principales (l'ACP) et l'auto-encodeur que nous détaillerons au Chapitre 2.

En résumé, deux approches principales sont proposées pour traiter nos problématiques :

- **l'approche *a posteriori*.** Le mot "*a posteriori*" s'entend posteriori par rapport à la construction du modèle. Dans ce mémoire, compte tenu du temps et des ressources limités, nous n'avons pas pu tester l'algorithme itératif mentionné au début de cette section. Pour y remédier, nous effectuerons les démarches suivantes :
  1. Un modèle de régression est calibré sur l'ensemble des données ;
  2. À partir du modèle construit, nous obtiendrons les prédictions de la variable à prédire et puis les résidus de déviance ;
  3. Une classification est effectuée sur ces résidus ;
  4. Un modèle de régression est calibré sur chaque groupe obtenu.
- **l'approche *a priori*,** c'est-à-dire qu'avant de construire un modèle, nous détectons les groupes homogènes en amont.

  Au sein de l'approche *a priori*, nous testerons différentes méthodes de modélisation dont la plupart se composent des étapes suivantes :
  1. préparation des données, notamment transformer les données qualitatives en quantitative ;

2. réduction de la dimension ;
3. classification. Plusieurs algorithmes permettent de la réaliser : K-means, SOM ... Il seront présentés au Chapitre 2 ;
4. régression : l'XGBoost réalisé groupe par groupe.

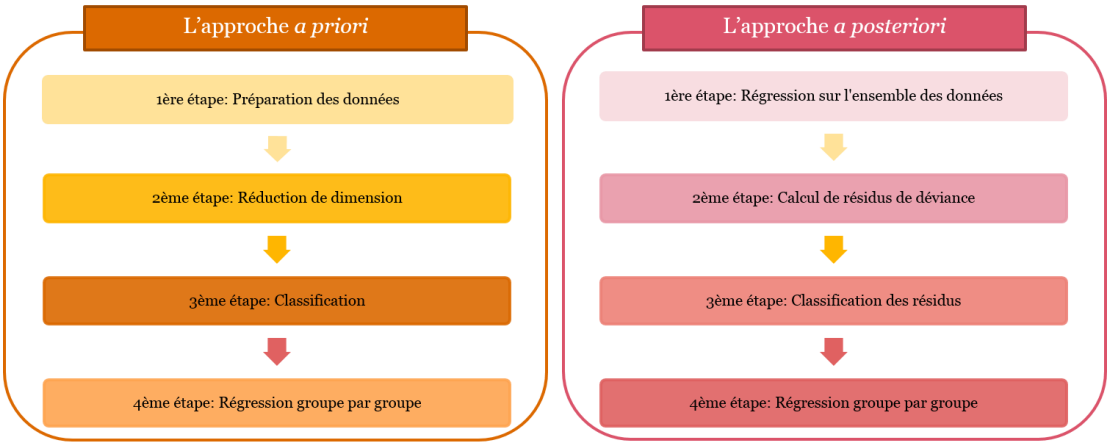

FIGURE 2.9 – Présentation générale des deux approches

Au Chapitre 2, la partie théorique de différentes méthodes à appliquer lors des différentes étapes sera présentée. Puis au Chapitre 3, ces méthodes seront mises en oeuvre afin de sélectionner une ou plusieurs méthodes adaptées à nos problématiques.

Chapitre 3

Apprentissage automatique : un pas dans la théorie

Le premier chapitre a montré le besoin d'identifier les groupes homogènes. Les méthodes de *Machine Learning*, ou algorithmes d'apprentissage automatique, pourront servir à résoudre nos problématiques. L'objectif de ce chapitre est d'introduire les notions et les théories nécessaires à la compréhension de nos méthodes. La première section de ce chapitre introduit les principes généraux des algorithmes d'apprentissage automatique. La deuxième section est consacrée aux algorithmes supervisés : CART et XGBoost. La troisième section présente les algorithmes non-supervisés. Nous y trouverons les algorithmes de réduction de la dimension (l'ACP et l'auto-encodeur) ainsi que ceux de clustering (K-means, SOM et GMM).

3.1 Notions générales des algorithmes d'apprentissage automatique

Algorithmes supervisés et algorithmes non-supervisés

Tout d'abord, parmi tous les algorithmes d'apprentissage automatique, il faut distinguer deux grandes catégories d'algorithmes : les algorithmes supervisés et les algorithmes non-supervisés :

- Dans le cas des algorithmes supervisés, nous disposons à la fois des variables explicatives, ou *features* en anglais et d'une variable à expliquer, ou variable cible. Le but de l'algorithme est de prédire la variable à expliquer à partir des variables explicatives.

- Dans le cas de ceux non-supervisés, les données que nous fournissons aux algorithmes sont les variables explicatives uniquement. Il n'y a pas de variable particulière à prédire.

Échantillon d'apprentissage et échantillon de test Avant le calibrage du modèle, la base de données considérée est toujours séparée en deux sous-échantillons :

- L'échantillon d'apprentissage, ou "*training set*" en anglais ;
- L'échantillon de test, ou "*test set*" en anglais.

Le premier, contenant souvent 80% ou 75% des données, servira à calibrer le modèle, c'est-à-dire que le modèle considéré va apprendre les paramètres à partir de l'échantillon d'apprentissage et déterminer les paramètres optimaux ; le dernier, contenant le reste des données, permettra de tester la performance du modèle construit. Plus précisément, pour une observation donnée de l'échantillon de test, nous cachons la valeur de la variable à prédire et nous demandons au modèle construit de la prédire à partir des variables explicatives. Nous comparons ensuite la valeur prédite avec la vraie valeur de la variable à prédire.

3.2 Algorithmes supervisés

En ce qui concerne les algorithmes supervisés, il y a plusieurs notions essentielles :

Notations Nous désignons $x_i \in \mathbb{R}^D$ la i-ème observation. Sa d-ème coordonnée vaut $x_i[d]$. Au total, nous avons N observations.

Modèle Un modèle est une fonction paramétrique ou non-paramétrique qui prend une observation x_i comme entrée et prédit $\hat{y}_i$.

Paramètres Parfois les paramètres d'un modèle sont appelés les poids ou les coefficients. Nous trouvons les paramètres optimisés à partir de nos données. Nous parlons alors d'apprentissage. Nous désignons Θ l'ensemble des paramètres d'un modèle. Par exemple, dans un modèle de régression linéaire, le modèle est $\hat{y}_i = \sum_{d=1}^{D} \omega_d x_i[d]$ et les paramètres sont $\Theta = \{\omega_d | d = 1, \ldots, D\}$.

Fonction objectif Une fonction objectif se compose souvent de deux parties : la fonction de perte et la fonction de régularisation :

$$Objectif(\Theta) = L(\Theta) + \Omega(\Theta)$$

La fonction de perte mesure la performance du modèle sur la base de données d'apprentissage. Par exemple, si nous prenons l'erreur quadratique, alors la fonction de perte s'écrira :

$$L(\Theta) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

La fonction de régularisation mesure la complexité du modèle. Afin d'éviter le sur-apprentissage ou *overfitting*, nous pénalisons les modèles trop complexes. Par exemple, si nous prenons la norme L2, alors la fonction de régularisation sera

$$\Omega(\Theta) = \lambda \sum_{d=1}^{D} \omega_d^2$$

où λ désigne l'hyper-paramètre qui contrôle le degré de pénalisation. Une grande valeur de λ pénalise beaucoup la complexité du modèle et inversement.

3.2.1 CART

L'algorithme CART est introduit par Breiman et col. (1984) dans le papier intitulé *Classification and Regression Trees*. En français nous parlons de l'arbre de classification et régression. C'est un algorithme supervisé qui permet de faire une classification ou une régression.

Comme dans ce mémoire, il est utilisé principalement dans le but de faire une régression, dans la suite, nous présenterons les principes généraux de cet algorithme en cas de régression. Il est à noter que plusieurs variantes de cet algorithme ont été proposées dans la littérature et ici nous présenterons une version simple sans perdre la généralité. Les lecteurs intéressés peuvent se référer au livre de Breiman. Tout d'abord, nous allons présenter quelques notions générales.

Fonction objectif

Comme tous les algorithmes supervisés, l'arbre de régression a aussi une fonction objectif à minimiser, qui est l'erreur quadratique moyenne :

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2,$$

où

— N est le nombre d'observations ;

— y_i est la vraie valeur de la variable à prédire ;

— $\hat{y}_i$ est la prédiction.

Construction de l'arbre

Toutefois l'algorithme n'optimise pas directement cette fonction objectif, en effet, l'optimisation se fait via un algorithme glouton (*greedy algorithm*). Considérons à présent la construction de l'arbre via l'algorithme glouton :

- Étape 1 : l'arbre a une racine contenant toutes les données. Nous commençons à la racine de l'arbre ;
- Étape 2 : L'algorithme sélectionne une variable qui partitionne le mieux les données dans deux groupes. "Le mieux" sera défini dans le paragraphe suivant ;
- Étape 3 : Les observations seront partitionnées ensuite en deux parties, ou deux "nœuds" selon la terminologie d'un arbre de décision. L'un de ces deux nœuds se nomme "le fils gauche" et l'autre "le fils droit" ;
- Étape 4 : Une fois que les données sont divisées en groupes, nous continuons à appliquer le même processus séparément à chaque nœud et ainsi de suite jusqu'à ce que nous arrivons à une feuille, i.e. un nœud sans fils où la condition d'arrêt est satisfaite. Nous parlons alors de partitionnement récursif.

Une fois l'arbre construit, nous pouvons faire les prédictions. En considérant chaque feuille comme un groupe, nous prenons la moyenne de la variable à prédire des observations au sein de chaque groupe pour la prédiction pour une nouvelle observation qui se trouve dans ce même groupe.

Le critère de partitionnement

Détaillons à présent le critère de partitionnement. i.e. à chaque nœud, afin de partitionner le mieux les données dans deux groupes, quelle variable explicative à choisir de façon optimale ? Une fois que nous avons décidé quelle variable explicative à utiliser, il y aura deux cas de figure suivants :

— Si nous avons choisi une variable quantitative, quelle valeur seuil sélectionner ?

— Si nous avons choisi une variable qualitative, quelles modalités regrouper ?

La réponse revient encore à optimiser la fonction objectif à chaque étape de la construction de l'arbre. Formalisons mathématiquement.

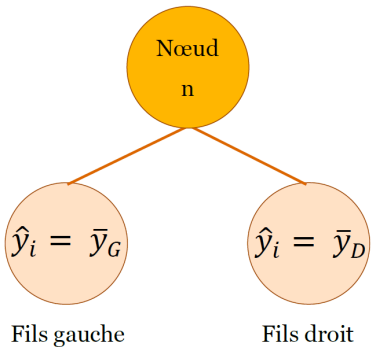

FIGURE 3.1

(Figure 3.1) Nous sommes au noeud n et nous souhaitons diviser les observations au sein de n en deux groupes. Notre but est toujours de minimiser la fonction objectif $\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$. A chaque noeud, la fonction objectif devient

$$\frac{|\mathcal{K}_G|}{N_n}\sum_{i\in\mathcal{K}_G}(y_i-\hat{y}_i)^2+\frac{|\mathcal{K}_D|}{N_n}\sum_{i\in\mathcal{K}_D}(y_i-\hat{y}_i)^2 = \frac{|\mathcal{K}_G|}{N_n}\sum_{i\in\mathcal{K}_G}(y_i-\bar{y}_G)^2+\frac{|\mathcal{K}_D|}{N_n}\sum_{i\in\mathcal{K}_D}(y_i-\bar{y}_D)^2$$

avec

— N_n : le nombre d'observations au nœud n ;

— $|\mathcal{K}_G|$: le nombre d'observations au fils gauche ;

— $|\mathcal{K}_D|$: le nombre d'observations au fils droit ;

— y_i : la vraie valeur de la variable à prédire ;

— $\hat{y}_i$: la prédiction pour l'observation i ;

— $\bar{y}_G$: la moyenne de la variable à prédire pour les observations au sein du fils gauche ;

— $\bar{y}_D$: la moyenne de la variable à prédire pour les observations au sein du fils droit.

L'algorithme pour déterminer la variable qui partitionne le mieux les données dans deux groupes s'écrit alors comme suit :

Étant donné l'ensemble de données au nœud n :

Pour chaque variable d, $d = 1, \ldots, p$:

Pour chaque modalité ou valeur de cette variable :

1) Partitionner les données selon la valeur ou les modalités de la variable d ;

2) Calculer la fonction objectif issue du partitionnement.

Choisir la variable d et le seuil ou les modalités de cette variable qui minimisent la fonction objectif.

Conditions d'arrêt

Parlons maintenant de l'autre notion importante de CART : les conditions d'arrêt. Le partitionnement s'arrête lorsque

— il n'y a plus d'amélioration même si nous continuons à partitionner les données en nœuds fils ;

— le nombre de données au sein des nœuds fils est inférieur à un seuil prédéfini, 5% de l'ensemble de données par exemple ;

— toutes les observations au sein d'un nœud ont la même valeur que la variable à prédire ;

— l'amélioration de la performance de prédiction est inférieure à un seuil définit *a priori*.

Élagage de l'arbre complet

Enfin pour éviter le problème de sur-apprentissage, en pratique, nous construisons d'abord un arbre complet et puis nous l'élaguons (en anglais, *prune*). Plus précisément, nous comparons la performance, i.e. la valeur de la fonction objectif du modèle en enlevant certaines feuilles, et celui en gardant ces feuilles sur un échantillon de validation. Puis nous décidons l'élagage selon les performances.

L'algorithme de CART nous permet d'avoir un partitionnement de l'espace relativement homogène vis-à-vis de la variable à prédire. L'arbre répartit les observations en groupes au sein desquels la variable à prédire prendra des valeurs assez proches. Nous supposons que les observations au sein d'un même groupe sont plus ou moins homogènes.

L'avantage de CART consiste au fait que les groupes identifiés à l'issue de cet algorithme se caractérisent facilement par les splits des variables. En effet, CART indique clairement l'endroit exact où il faut couper une variable. Comparé à CART, d'autres algorithmes de classification non-supervisée comme K-means, SOM, GMM ne possèdent pas cette propriété et donc il s'avère difficile de caractériser les groupes identifiés par K-means ou GMM.

3.2.2 XGBoost

L'XGBoost est l'un des plus puissants et populaires algorithmes supervisés. Dans de nombreuses compétitions de *data-science*, notamment celles de *Kaggle* , les vainqueurs ont gagné les concours grâce à l'XGBoost. Citons les mots d'un ex-Numéro 1 de *Kaggle* :

> *When in doubt, use XBGoost.* [1]
>
> — Owen Zhang, ex-Numéro 1 de *Kaggle*

L'XGBoost a gagné sa popularité notamment pour les raisons suivantes :

- Il est facile à implémenter : contrairement à beaucoup d'autres méthodes par exemple celles basées sur les modèles linéaires, il n'est pas toujours nécessaire de vérifier telle ou telle hypothèse particulière.
- Il est beaucoup plus rapide que d'autres algorithmes.
- L'essentiel est que sa performance est mille fois meilleure que les autres en terme de MSE, MAE lorsqu'il s'agit de régression.

Un exemple simpliste de "*Gradient Boosting Machine*"

L'XGBoost est l'abréviation pour "*eXtreme Gradient Boosting*" en anglais. Comme son nom l'indique, c'est une variante de "*Gradient Boosting Machine*" qui est introduite par Friedman dans son papier *Greedy Function Approximation : A*

1. Traduction en français : *Dans le doubt, utiliser XGBoost.*

Gradient Boosting Machine. L'idée de *boosting* est très simple : est-ce que nous pouvons faire mieux avec un "*weak learner*" ? La réponse est oui. Nous pouvons améliorer la performance de l'algorithme en combinant un ensemble de "*weak learner*" séquentiellement.

Voici un exemple simple[2] d'un *Gradient Boosting Machine*. Considérons un problème de classification : il y a deux classes, dont une est désignée par les cercles de couleur bleue et l'autre par les triangules oranges. Le but consiste à correctement distinguer l'appartenance à telle ou telle classe d'une observation donnée.

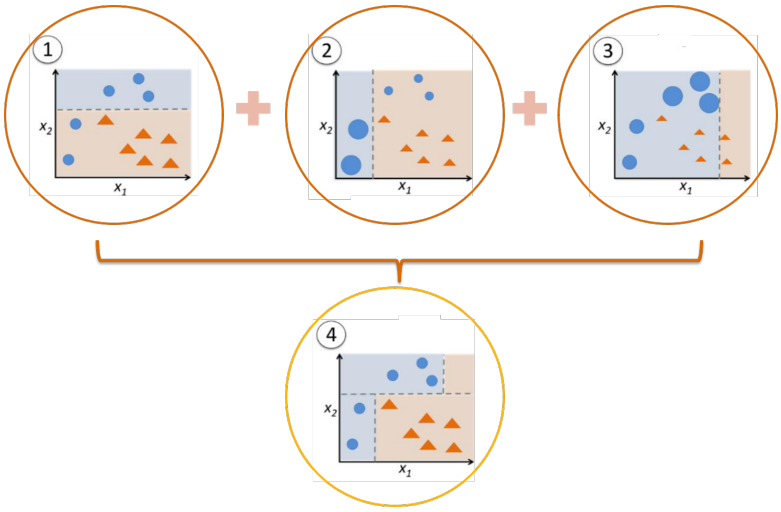

FIGURE 3.2 – Illustration d'un *Gradient Boosting Machine*

Supposons qu'à l'itération 1, notre règle de décision est : étant donnée une observation, nous nous donnons un seuil de x_2, si x_2 de l'observation concernée est supérieur à ce seuil, nous la classons en cercle bleu ; si son x_2 est inférieur à ce seuil, nous la classons en triangle orange. Comme la règle de décision est très simple et la performance de l'algorithme n'est pas satisfaisante, nous parlons alors de "*weak learner*". À la sortie de l'algorithme, nous remarquons que les deux cercles bleus en bas à gauche sont mal classés : nous les avons répertoriés en triangles oranges selon notre règle de décision, mais en réalité ils sont de classe cercle bleu. Les restes des observations sont tous correctement classés.

À l'itération 2, afin de corriger les erreurs commises à l'itération 1, nous donnons plus de poids aux deux observations mal classées et moins de poids aux

2. Source de la figure : Raschka, *Pythnous Machine Learning*

restes. Notre deuxième règle de décision sera alors : nous nous donnons un seuil à la valeur x_1, si x_1 de l'observation concernée est inférieure à ce seuil, alors nous la classons en cercle bleu ; sinon, nous la classons en triangle orange. À l'issue de cette règle de décision, les trois observations en haut à droite sont mal classées : elles sont classées en triangle orange mais en réalité elles appartiennent à la classe cercle bleu.

Il en est de même pour l'itération 3. À l'itération 3, afin de corriger les erreurs commises à l'itération 2, nous donnons plus de poids aux trois observations mal classées et moins de poids aux restes. Notre troisième règle de décision sera alors : nous nous donnons un seuil à la valeur de x_1, si x_1 de l'observation concernée est inférieure à ce seuil, alors nous la classons en cercles bleu ; sinon, en triangles oranges.

Enfin, nous agrégeons toutes les règles de décision simples, c'est-à-dire que nous faisons une combinaison linéaire des règles de décision simples en donnant plus de poids à celles qui prédisent bien et moins de poids à celles qui prédisent mal. Cela donne le résultat final de la Figure 3.2.

L'algorithme d'XGBoost

Formalisons mathématiquement l'algorithme de XGBoost.

Dans l'algorithme XGBoost, un "*weak learner*" est un arbre de décision. Nous désignons l'arbre de la k-ème itération f_k, $k = 1, \ldots, K$ et le modèle final d'agrégation s'écrit

$$\sum_{k=1}^{K} f_k$$

Alors pour une observation x_i donnée, la prédiction sera

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i).$$

Un concept essentiel de l'algorithme d'XGBoost consiste au fait que l'apprentissage se fait de manière séquentielle, c'est-à-dire que le modèle final $\sum_{k=1}^{K} f_k$ n'est pas calibré d'un coup. En effet, nous apprenons les f_k l'un après l'autre. Plus précisément, à l'initialisation, nous prédisons que la variable à prédire vaut une constante, disons 0 :

$$\hat{y}_i^{(0)} = 0.$$

À l'itération 1, la prédiction de la variable à prédire est donnée par

$$\hat{y}_i^{(1)} = \hat{y}_i^{(0)} + f_1(x_i)$$

et nous n'avons donc que f_1 à apprendre.
À l'itération 2, la prédiction de la variable à prédire est donnée par

$$\begin{aligned}\hat{y}_i^{(2)} &= \sum_{k=1}^{2} f_k(x_i) \\ &= \hat{y}_i^{(1)} + f_2(x_i)\end{aligned}$$

où $\hat{y}_i^{(1)}$ est déjà connu grâce aux itérations précédentes, et nous n'avons donc que f_2 à apprendre.
Ainsi de suite. À l'itération t,

$$\begin{aligned}\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) \\ &= \hat{y}_i^{(t-1)} + f_t(x_i)\end{aligned}$$

et nous calibrons f_t.

Comme la plupart des algorithmes que nous connaissons,la fonction objectif de l'XGBboost se compose de deux parties : la fonction de perte et la fonction de régularisation :

$$\begin{aligned}Objectif(\Theta) &= L(\Theta) + \Omega(\Theta) \\ &= \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k).\end{aligned}$$

À l'itération t, comme nous n'avons que f_t à apprendre, la fonction objectif devient alors

$$Objectif^{(t)}(\Theta) = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t)}) + \sum_{k=1}^{t} \Omega(f_k)$$

Comme

$$\begin{aligned}\hat{y_i}^{(t)} &= \sum_{k=1}^{t} f_k(x_i) \\ &= \sum_{k=1}^{t-1} f_k(x_i) + f_t(x_i) \\ &= \hat{y_i}^{(t-1)} + f_t(x_i)\end{aligned}$$

En remplaçant $\hat{y_i}^{(t)}$ par sa valeur, nous pouvons réécrire la fonction objectif de la façon suivante

$$Objectif^{(t)}(\Theta) = \underbrace{\sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i))}_{\text{fonction de perte}} + \underbrace{\sum_{k=1}^{t} \Omega(f_k)}_{\text{fonction de régularisation}} \quad .$$

Considérons la fonction de perte $\sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i))$.
Le développement limité à l'ordre 2 nous donne

$$l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) \approx l(y_i, \hat{y_i}^{(t-1)}) + \frac{\partial l(y_i, \hat{y_i}^{(t-1)})}{\partial \hat{y_i}^{(t-1)}} \cdot f_t(x_i) + \frac{1}{2} \frac{\partial^2 l(y_i, \hat{y_i}^{(t-1)})}{\partial (\hat{y_i}^{(t-1)})^2} \cdot (f_t(x_i))^2.$$

Nous désignons $\frac{\partial l(y_i, \hat{y_i}^{(t-1)})}{\partial \hat{y_i}^{(t-1)}}$ par g_i et $\frac{\partial^2 l(y_i, \hat{y_i}^{(t-1)})}{\partial (\hat{y_i}^{(t-1)})^2}$ par h_i, alors la fonction de perte s'écrit

$$\sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) \approx \sum_{i=1}^{n} \underbrace{l(y_i, \hat{y_i}^{(t-1)})}_{\text{constant}} + g_i \cdot f_t(x_i) + \frac{1}{2} h_i \cdot (f_t(x_i))^2 \qquad (3.1)$$

Maintenant, considérons l'autre partie : la fonction de régularisation. f_t étant un arbre, nous définissons cet arbre par deux éléments :

— Un vecteur de "scores" : chaque feuille possède un score. Les "scores" sont analogues aux coefficients dans un modèle de régression linéaire.

— La structure de l'arbre, plus précisément, une fonction qui affecte les observations aux feuilles.

Formellement, un arbre peut se définir comme suit

$$f_t(x) = \omega_{q(x)}$$

avec

— x une observation donnée

— T le nombre de feuilles de l'arbre

— $\omega \in \mathbb{R}^{\mathrm{T}}$: le "score" d'une feuille. Nous attribuons le même score à toutes les observations au sein d'une même feuille.

— q une fonction qui affecte x à une certaine feuille : $\mathbb{R}^{\mathrm{d}} \longrightarrow \{1, ..., \mathrm{T}\}$

Cette définition nous dit qu'une prédiction via un arbre se fait en deux étapes :

1. Nous affectons l'observation x à une feuille via la fonction q.
2. Nous attribuons le score de $q(x)$-ième feuille à cette observation.

La complexité de l'itération t se définit

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \tag{3.2}$$

où

— T désigne le nombre de feuilles de l'arbre ;

— $\sum_{j=1}^{T} \omega_j^2$ est la norme $L2$ des "scores" ;

— γ et λ sont respectivement les "hyperparamètres" qui contrôlent le degré de pénalisation. Une grande valeur de γ ou λ pénalise beaucoup la complexité des scores.

Nous définissons aussi l'ensemble des observations au sein de la feuille j :

$$I_j = \{i | q(x_i) = j\}$$

où x_i désigne l'observation i

La fonction objectif à l'itération t s'écrit alors comme la somme de (1) et (2) :

$$\begin{aligned} Objectif^{(t)} &\approx \underbrace{\sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)}) + g_i \cdot f_t(x_i) + \frac{1}{2} h_i \cdot (f_t(x_i))^2}_{L(\Theta)} + \underbrace{\gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2}_{\Omega(\Theta)} \\ &= \underbrace{\sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)})}_{\text{constant}} + \sum_{j=1}^{T} \left\{ \Big(\sum_{i \in I_j} g_i \Big) \omega_j + \Big(\frac{1}{2} \sum_{i \in I_j} h_i \Big) \omega_j^2 \right\} + \gamma T + \sum_{j=1}^{T} \frac{1}{2} \lambda \omega_j^2 \\ &= \sum_{j=1}^{T} \underbrace{\left\{ \Big(\sum_{i \in I_j} g_i \Big) \omega_j + \frac{1}{2} \Big(\sum_{i \in I_j} h_i + \lambda \Big) \omega_j^2 \right\}}_{\text{à étudier}} + \gamma T + \underbrace{\sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)})}_{\text{constant}} \end{aligned}$$

Nous allons étudier les termes dans cette somme.
Notons $G_j = \sum_{i \in I_j} g_i$ et $H_j = \sum_{i \in I_j} h_i$
Alors

$$\Big(\sum_{i \in I_j} g_i \Big) \omega_j + \frac{1}{2} \Big(\sum_{i \in I_j} h_i + \lambda \Big) \omega_j^2 = \frac{1}{2} (H_j + \lambda) \omega_j^2 + G_j \omega_j$$

C'est une fonction quadratique en ω_j. Son minimum est atteint en $\omega_j^* = -\frac{G_j}{H_j + \lambda}$ et lorsque ce minimum est atteint, il vaut $-\frac{G_j^2}{2(H_j + \lambda)}$. Ainsi pour chaque feuille, le score optimal vaut ω_j^*.

Si nous supposons que la structure de l'arbre est fixée, i.e. nous connaissons le nombre de feuilles de l'arbre et la fonction q, alors en prenant le score optimal de chaque feuille, notre fonction objectif pour l'itération t s'écrit

$$Objectif^{(t)} = \sum_{j=1}^{T} -\frac{G_j^2}{2(H_j + \lambda)} + \gamma T + \underbrace{\sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)})}_{\text{constant}}$$

À présent, il reste encore une dernière question : comment est-ce que nous déterminons la structure de l'arbre q ? L'idée principale ressemble à l'algorithme de CART. Comme nous avons déjà présenté CART dans la section précédente, nous ne la détaillerons pas plus ici. Toutefois, il convient de remarquer que dans l'algorithme d'XGBoost, un algorithme glouton (greedy algorithme) est implémenté afin de déterminer la structure optimale d'un arbre. Les lecteurs intéressés sont

invités à consulter de nombreuses références disponibles, notamment la documentation en ligne de l'XGBoost, pour approfondir son étude.

3.3 Algorithmes non-supervisés

3.3.1 Réduction de la dimension

Lorsque nous avons une base de données de grande dimension avec 100 voire 1000 variables explicatives, il est nécessaire de réduire la dimension de données. Ceci est principalement dû au "fléau de la grande dimension", ou "*curse of dimensionality*" en anglais. En fait, lorsque la dimension augmente, les données deviennent isolées et éparses. Par conséquent, les algorithmes de clustering comme K-means que nous allons détailler plus tard ne fonctionnent pas très bien en grande dimension.

Certes, nous risquons de perdre une partie d'information suite à la réduction de la dimension, tout comme lorsque nous compressons une image en format JPEG, la qualité de l'image peut se dégrader. Toutefois, les méthodes de réduction de la dimension permettront non seulement de trouver une nouvelle représentation des données dans un espace de dimension beaucoup plus petite que celle de l'espace d'origine, mais aussi de conserver la plupart des informations essentielles.

3.3.1.1 Analyse en composantes principales

L'analyse en composantes principales (l'ACP) est probablement l'une des méthodes les plus utilisées pour réduire la dimension. D'une manière générale, il s'agit de trouver un espace de faible dimension engendré par plusieurs "axes" tel que la projection des données d'origine sur cet espace conservent le maximum de variance.

Sans perte de généralité, prenons un exemple en dimension 2. Nous souhaitons réduire la représentation des données en dimension 1. Pour l'analyse en composantes principales, cela signifie projeter les observations de dimension 2 dans un espace de dimension 1. Plus précisément, il faut trouver un vecteur, noté $u^{(1)}$, tel que : en projetant les observations sur $u^{(1)}$, nous déformons les observations le moins possible, en sens de l'ACP, c'est-à-dire en préservant le maximum de variance.

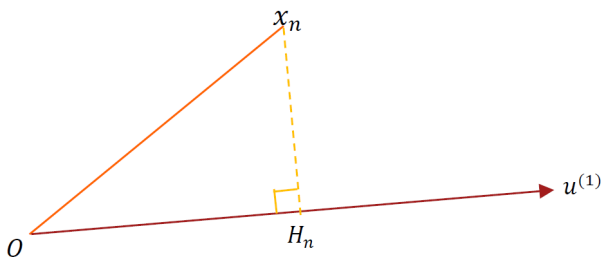

FIGURE 3.3 – Illustration de l'ACP en dimension 2

Soit x_n une observation centrée et réduite quelconque. Nous notons sa projection sur $u^{(1)}$ H_n (Figure 3.3). Comme x_n est centrée et réduite, le point O correspond à l'origine. Nous souhaitons alors que sa projection OH_n soit la plus grande que possible. Comme maximiser $\|OH_n\|$ est équivalent à maximiser $\|OH_n\|^2$ et de plus nous supposons qu'au total, nous disposons de N lignes d'observations ayant toutes le même poids. La fonction objectif à maximiser s'écrit alors

$$\sum_{n=1}^{N} \|OH_n\|^2$$

Nous parlons alors de variance ou inertie maximum.

De manière plus générale, si nous souhaitons réduire la représentation des données de dimension p à dimension k, il faut trouver k vecteurs $u^{(1)}, \ldots, u^{(k)}$. L'espace formé par ces vecteurs sera le nouvel espace sur lequel nous travaillerons. Nous obtiendrons la nouvelle base de données en dimension k en projetant les observations d'origine sur ce nouvel espace avec l'inertie maximum.

Comment trouvons-nous ces k vecteurs ?

- Étape 0 : Avant que nous procédons à l'ACP, il est toujours nécessaire de centrer et réduire les données afin d'éliminer l'effet des unités de mesure et supprimer les effets dus aux ordres de grandeur différents. Dans la suite, nous notons l'ensemble de données centrées et réduites la matrice X, qui est composée de N lignes et p colonnes. Chaque ligne de X correspond à une observation alors que chaque colonne de X correspond à une variable.
- Étape 1 : Nous calculons la matrice de corrélation $\Sigma = \frac{1}{N} X^T X$.

- Étape 2 : Nous calculons les valeurs propres et les vecteurs propres de cette matrice Σ. nous notons les valeurs propres $\lambda_1, \ldots \lambda_p$, avec $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_p$. Les vecteurs propres correspondants sont notés $u^{(1)}, \ldots, u^{(p)}$.
- Étape 3 : Comme nous souhaitons réduire les observations de dimension p à la dimension k, nous prenons alors les k premiers vecteurs propres : $u^{(1)}, \ldots, u^{(k)}$. Ces vecteurs propres sont appelés "axes principaux" ou "composantes principales". Nous pouvons montrer qu'ils sont en fait les combinaisons linéaires des variables initiales. Nous définissons ensuite la matrice U dont les colonnes sont ces k vecteurs propres.
- Étape 4 : Les nouvelles coordonnées issues de la projection des observations sur le nouvel espace engendré par ces vecteurs propres sont données par $Z = X \cdot U$, qui est de N lignes et de k colonnes.

Nous pouvons montrer que les vecteurs propres ainsi trouvés sont tels que l'inertie soit maximum. De plus, ils sont orthogonaux entre eux.

Il convient de remarquer que l'analyse en composantes principales possède d'autres propriétés importantes notamment dans l'analyse des données que nous ne détaillerons pas dans ce mémoire. Notons juste que lorsque la structure de données est complexe, nous risquons de perdre beaucoup d'information en réduisant la dimension via l'ACP.

3.3.1.2 Réseau de neurones artificiels et auto-encodeur

L'autre méthode de réduction de la dimension est l'auto-encodeur. Avant de présenter l'auto-encodeur, nous allons introduire l'algorithme de réseau de neurones artificiels d'abord car l'auto-encodeur n'est en effet qu'un type particulier de réseau de neurones.

Réseau de neurones artificiels

Présentation d'un neurone

L'algorithme de réseau de neurones artificiels est inspiré par les neurones biologiques. Tout comme un neurone biologique, dans l'algorithme, chaque neurone reçoit des signaux d'entrée, procède les informations reçues puis son résultat de sortie sera l'entrée d'autres neurones.

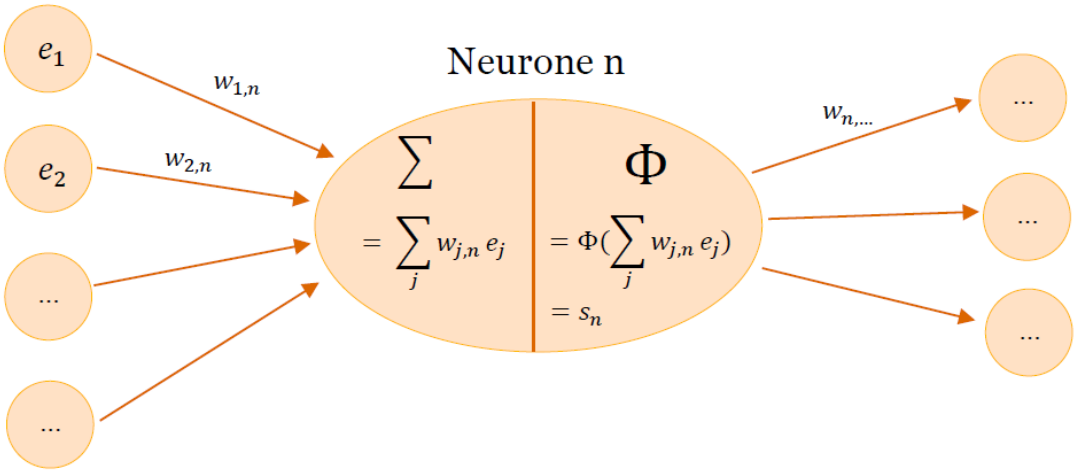

FIGURE 3.4 – Présentation d'un neurone

La Figure 3.4 montre le travail d'un neurone noté n :

- e_1 désigne la valeur d'entrée numéro 1 ;
- e_2 désigne la valeur d'entrée numéro 2, etc.
- $w_{1,n}$ désigne le poids reliant l'entrée numéro 1 et le neurone n ;
- $w_{2,n}$ désigne le poids reliant l'entrée numéro 2 et le neurone n, etc.

Le neurone n, dans un premier temps, calcule la combinaison linéaire des valeurs d'entrée : $\sum_j w_{j,n}e_j$; puis dans un deuxième temps il passe cette somme à une fonction d'activation Φ. La sortie de ce neurone vaut alors $\Phi(\sum_j w_{j,n}e_j)$. Cette sortie devient l'entrée d'autres neurones.

D'un neurone à un réseau de neurones

Comme son nom l'indique, l'algorithme de réseau de neurones est composé d'un ensemble de neurones.

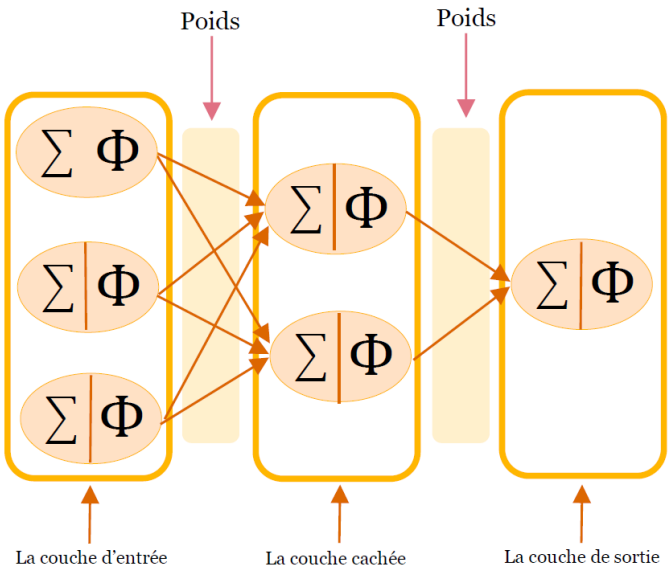

FIGURE 3.5 – Présentation d'un réseau de neurones

La Figure 3.5 montre un réseau de neurones de version simpliste mais sans perdre la généralité qui comprend les parties suivantes :

- **La couche d'entrée**, ou "*input layer*", comme son nom l'indique, est l'entrée du réseau de neurones, où nous fournissons les données d'entrée à notre modèle.
- **Les couches cachées**, ou "*hidden layers*". Toutes les couches entre la couche d'entrée et celle de sortie sont appelées "les couches cachées".
- **La couche de sortie**, ou "*output layer*", est la dernière couche où se trouvent les résultats de l'estimation.

Pour chaque couche (sauf la couche d'entrée), chaque neurone prend ses entrées sur les sorties des neurones dans la couche précédente. Nous parlons alors de "Forward propagation". Ces entrées sont associées avec des poids différents. L'apprentissage de réseau de neurones consiste précisément à trouver les poids optimaux.

L'apprentissage de réseau de neurones : rétro-propagation

Comment trouvons-nous les poids optimaux ? En 1986, D.Rumelhart, G. Hinton et al. ont publié un article intitulé *Learning Internal Representation by Error*

Propagation qui est une percée majeure. Dans cet article ils ont introduit l'algorithme de "rétro-propagation", ou "*backpropagation*" en anglais. Cet algorithme permet de trouver les poids efficacement.

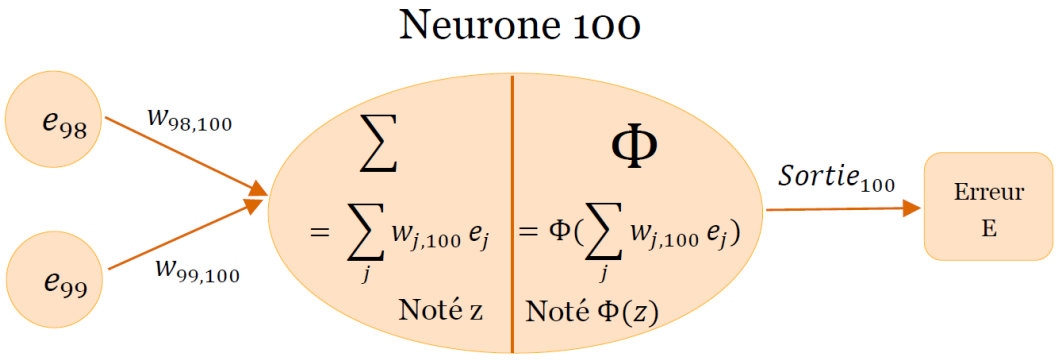

FIGURE 3.6

Considérons un exemple simple. Voici le neurone 100 (Figure 3.6), qui prend e_{98} et e_{99} comme entrées. Sa sortie, noté $Sortie_{100}$ sera la prédiction. Nous les avons numérotés comme $100, 98, 99$ pour facilement repérer les neurones.

Pour une observation donnée, la fonction objectif à minimiser s'écrit

$$E = \frac{1}{2}(y - Sortie_{100})^2$$

où y désigne la vraie valeur de la variable à prédire.

Par ailleurs, afin de faciliter le calcul par la suite nous supposons que la fonction d'activation est la fonction sigmoïde qui se définit par :

$$\Phi(z) = \frac{1}{1 + e^{-z}}$$

Calculons la dérivée partielle de la fonction objectif en fonction du poids $w_{98,100}$:

$$\frac{\partial E}{\partial w_{98,100}} = \frac{\partial E}{\partial Sortie_{100}} \frac{\partial Sortie_{100}}{\partial z} \frac{\partial z}{\partial w_{98,100}}$$

Or, le premier terme vaut

$$\begin{aligned}\frac{\partial E}{\partial Sortie_{100}} &= \frac{\partial}{\partial Sortie_{100}}\frac{1}{2}(y - Sortie_{100})^2 \\ &= \frac{1}{2}\cdot 2\cdot(y - Sortie_{100})\cdot(-1) \\ &= -(y - Sortie_{100})\end{aligned}$$

Le deuxième terme vaut

$$\begin{aligned}\frac{\partial Sortie_{100}}{\partial z} &= \frac{\partial \Phi(z)}{\partial z} \\ &= \Phi(z)(1 - \Phi(z)) \\ &= Sortie_{100}(1 - Sortie_{100})\end{aligned}$$

Le troisième terme vaut

$$\begin{aligned}\frac{\partial z}{\partial w_{98,100}} &= \frac{\partial(w_{98,100}e_{98} + w_{99,100}e_{99})}{\partial w_{98,100}} \\ &= e_{98}\end{aligned}$$

En rassemblant tous les trois termes, nous obtenons

$$\frac{\partial E}{\partial w_{98,100}} = -(y - Sortie_{100})\cdot Sortie_{100}(1 - Sortie_{100})\cdot e_{98}$$

Nous définissons δ_{100} comme $-(y - Sortie_{100})\cdot Sortie_{100}(1 - Sortie_{100})$ car ce terme ne dépend que du neurone 100.

Jusqu'à maintenant, nous n'avons fait des calculs que sur la dernière couche cachée. Nous allons étudier la couche précédente. Pour être cohérent, nous gardons les mêmes notations et pour des raisons de commodité, nous nommons tous les neurones de la dernière couche cachée "neurones 90's" et ceux de la couche précédente "neurones 80's" pour que nous puissons repérer les neurones facilement (Figure 3.7 et Figure 3.8).

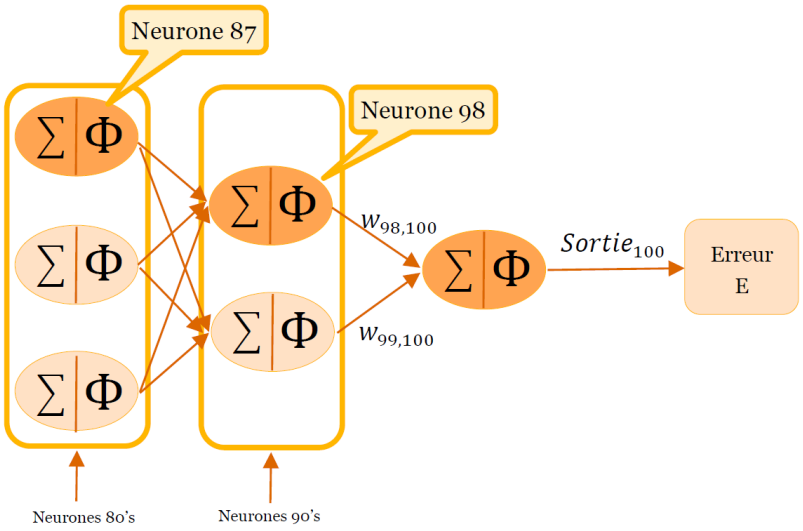

FIGURE 3.7

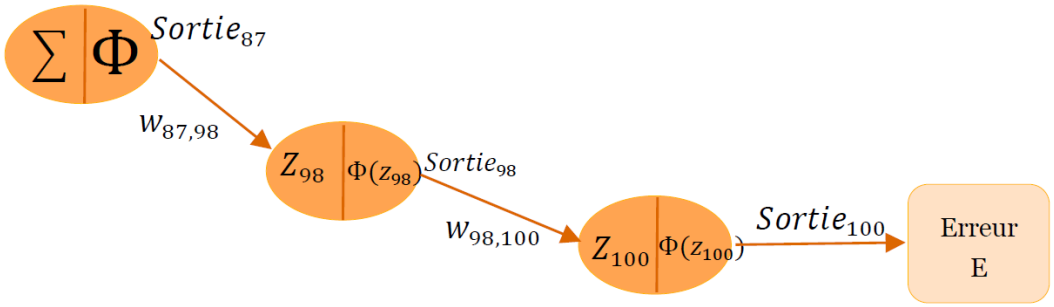

FIGURE 3.8

Calculons la dérivée partielle de la fonction objectif en fonction du poids $w_{87,98}$:

$$\frac{\partial E}{\partial w_{87,98}} = \frac{\partial E}{\partial Sortie_{98}} \frac{\partial Sortie_{98}}{\partial z_{98}} \frac{\partial z_{98}}{\partial w_{87,98}}$$

Or, le premier terme vaut

$$\begin{aligned} \frac{\partial E}{\partial Sortie_{98}} &= \frac{\partial E}{\partial z_{100}} \frac{\partial z_{100}}{\partial Sortie_{98}} \\ &= \delta_{100} \cdot w_{87,98} \end{aligned}$$

Le deuxième terme vaut

$$\begin{aligned}\frac{\partial Sortie_{98}}{\partial z_{98}} &= \frac{\Phi(z_{98})}{z_{98}} \\ &= \Phi(z_{98})(1-\Phi(z_{98})) \\ &= Sortie_{98}(1-Sortie_{98})\end{aligned}$$

Le troisième terme vaut

$$\frac{\partial z_{98}}{\partial w_{87,98}} = Sortie_{98}$$

Enfin, en rassemblant tous les termes, nous obtenons

$$\frac{\partial E}{\partial w_{87,98}} = \delta_{100} \cdot w_{87,98} \cdot Sortie_{98} \cdot (1-Sortie_{98}) \cdot Sortie_{98}$$

De même, nous calculons les dérivées partielles de la fonction objectif en fonction des poids de la dernière couche cachée ; puis celles de l'avant dernière couche cachée ; ensuite celles de la troisième avant dernière couche cachée, ainsi de suite, jusqu'à que nous arrivons à la couche d'entrée. Nous obtenons toutes les dérivées partielles $\frac{\partial E}{\partial w_{i,j}}$ où i, j désignent respectivement les numéros de neurones. Nous pouvons alors obtenir les poids optimaux via un algorithme de descente de gradient :

$$w_{i,j} \leftarrow w_{i,j} - \alpha \frac{\partial E}{\partial w_{i,j}}$$

où α désigne la vitesse d'apprentissage, ou "*learning rate*" en anglais.

Il convient de remarquer que l'algorithme de descente de gradient ne garantit qu'une convergence à un minimum local. D'autres variantes sont développées pour y remédier, citons l'algorithme de gradient de descente stochastique (SGD) pour mémoire. Ces algorithmes sont implémentés dans la plupart des packages ou logiciels de réseau de neurones. Compte tenu que le sujet central de cette partie est l'introduction de l'algorithme de réseau de neurones artificiel, nous n'approfondirons donc pas plus sur le sujet.

Concisément, les paramètres de l'algorithme de réseau de neurones sont estimés par "*forward propagation*" et "rétro-propagation". Après une initialisation des poids d'une manière aléatoire, à chaque itération, les neurones de la couche d'entrée lisent une observation ; puis nous parcourons des neurones couche par couche de la première couche cachée jusqu'à la dernière couche qui est la couche de sortie, et fait une prédiction. Nous parlons alors de "*forward propagation*". Ensuite, l'erreur est calculée en comparant cette prédiction avec la vraie valeur de la variable à prédire ; puis nous revenons à l'avant en passant par chaque couche pour ajuster les poids. Il s'agit alors de "rétro-propagation".

L'algorithme de réseau de neurones est très performant, et est souvent utilisé par exemple dans les domaines de reconnaissance d'image, de son.

Du réseau de neurones à l'auto-encodeur

L'un des points forts de l'algorithme de réseau de neurones consiste au fait que l'algorithme apprend progressivement les représentations de données via "*forward propagation*". L'auto-encodeur n'est en fait qu'un type particulier de réseau de neurones. Il est capable d'apprendre des représentations synthétiques des données et capter des structures spécifiques.

Prenons un exemple simple. Voici deux séries de nombres à mémoriser :

- La première étant 37, 23, 45, 34, 99, 10
- La deuxième étant 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

La première série a l'air d'être beaucoup plus facile à mémoriser que la dernière tout simplement parce qu'elle contient moins de chiffres ! Toutefois si nous étudions consciencieusement la dernière, nous remarquerons qu'il y a un "pattern" qui se cache derrière. En fait, dans la deuxième série, si c'est un nombre pair, nous le divisons par deux et nous obtenons le nombre suivant ; si c'est un nombre impair, alors nous le multiplions par trois et en ajoute un. Une fois que nous avons découvert cette topologie, cette série devient trop facile à mémoriser ! L'auto-encodeur fonctionne en quelque sorte de la même manière.

L'idée de l'auto-encodeur consiste à prendre les données d'entrée de dimension p puis dans un premier temps les compresser en k dimensions avec $k < p$. Cette

partie est appelée “encodage” et les données compressées sont appelées“codes”. Dans un second temps, nous reprenons les données compressées en dimension k et nous essayons de les transformer en données de restitution de dimension p qui ressemblent le plus possible aux données initiales. Cette partie s’appelle “décodage”. Voici une illustration :

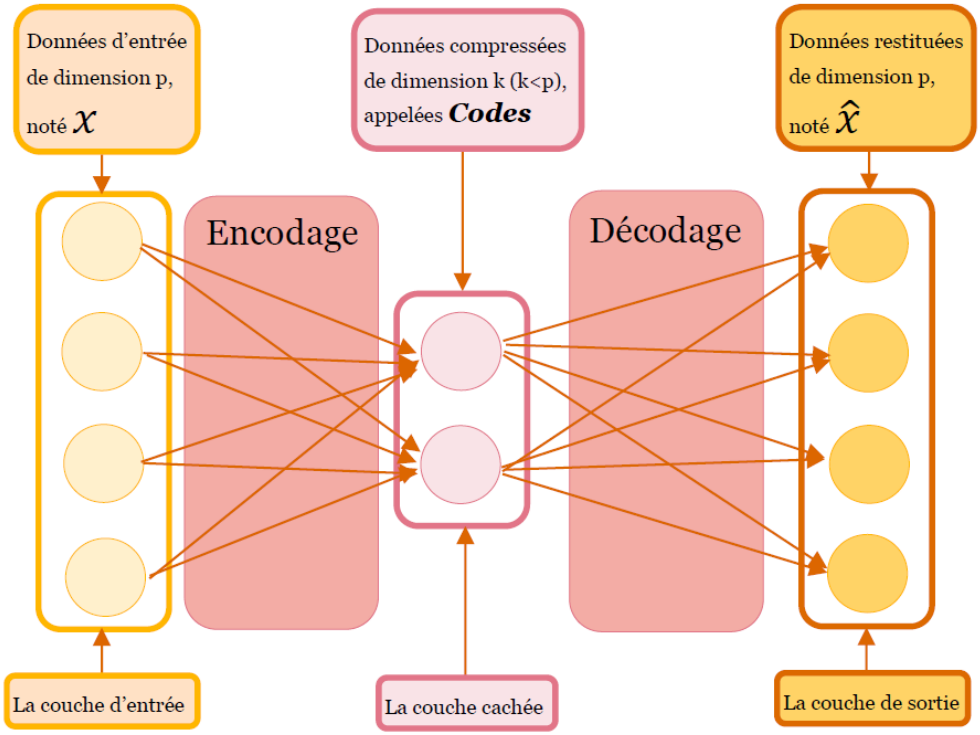

FIGURE 3.9 – Illustration d’un auto-encodeur

Comme nous souhaitons reconstruire les données d’entrée, pour une observation x donnée, il parait naturel de définir l’erreur de reconstruction qui permet de mesurer combien la reconstruction est proche de l’observation d’entrée. L’erreur de reconstruction se définit comme :

$$\|x - \hat{x}\| = \sqrt{\sum_{d=1}^{p} (x[d] - \hat{x}[d])^2}$$

où x désigne le vecteur d’entrée pour une observation quelconque, $\hat{x}$ le vecteur de sortie, $x[d]$ la d-ième coordonnée de x, $\hat{x}[d]$ la d-ième coordonnée de $\hat{x}$.

Nous prendrons ensuite la moyenne de cette quantité sur l'ensemble de l'échantillon. Ainsi la fonction objectif à minimiser s'écrit

$$\frac{1}{N}\sum_{i=1}^{N}\|x_i - \hat{x}_i\|$$

où N désigne le nombre d'échantillons et x_i l'observation numéro i.

Les principes de l'algorithme de l'auto-encodeur sont exactement identiques au réseau de neurones introduit précédemment. Les seules différences résident dans le fait que dans un réseau de neurones typique, la sortie est la valeur à prédire alors que dans un auto-encodeur, la sortie est $\hat{x}$, la restitution des données d'entrée.

Le *benchmark* de l'auto-encodeur est l'exemple que Hinton a donné dans son article *Reducing the dimentionality of data with neural networks* en 2006.

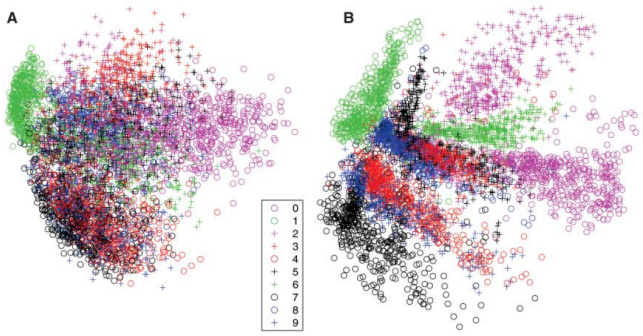

FIGURE 3.10 – Le *benchmark* de l'auto-encodeur

Dans cet exemple, le but est d'identifier les digits $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$ étant donné un chiffre manuscrit. Avec le résultat de l'analyse en composantes principales en dimension 2 (Figure 3.10 A), il nous parait très difficile d'identifier les groupes alors qu'avec le résultat de l'auto-encodeur (Figure 3.10 B), Nous pouvons facilement appliquer un autre algorithme de classification non-supervisée comme K-means et distinguer les groupes facilement.

Dans certains cas, s'il n'y a qu'une couche cachée, la fonction d'activation étant l'identité, la fonction objectif étant l'erreur quadratique, Nous pouvons montrer

que le résultat de l'auto-encodeur est identique à celui de l'analyse en composantes principales. Or, grâce aux couches cachées et aux fonctions d'activation, l'auto-encodeur s'avère beaucoup plus puissant que l'analyse en composantes principales. En effet, comme la dimension des "codes" est beaucoup réduite comparée à celle des données d'entrée, l'auto-encodeur est forcé d'apprendre les caractéristiques des variables les plus importantes, extraire les informations les plus essentielles et les condenser dans "codes". Il en résulte que les "codes" sont comme des variables synthétiques, permettant au mieux de résumer l'information contenue dans les données.

3.3.2 Classification non-supervisée

La classification non-supervisée, ou *clustering* en anglais, consiste à chercher une structure intrinsèque des données et détecter les groupes similaires tels que :

- Premièrement, les observations au sein d'un même groupe se ressemblent entre elles ;
- Deuxièmement, les observations appartenant à différents groupes se distinguent entre elles le plus possible.

Étant donné cet objectif, la notion de ressemblance ou similarité est alors essentielle pour les algorithmes de classification non-supervisée. Nous pouvons définir la notion de similarité par sa notion duale de dissemblance. En mathématiques, la dissemblance est souvent quantifiée par distance. La distance indique le degré de dissemblance entre les observations. Plus la distance entre observations est grande, plus les observations se distinguent entre elles. La plupart des méthodes de classification non-supervisée reposent sur la distance euclidienne.

Afin d'évaluer la qualité de *clustering*, il est alors nécessaire de définir les mesures d'évaluation. Deux notions importantes sont souvent évoquées dans la littérature du *clustering* :

- **Inertie intra-classe**. L'homogénéité des observations au sein d'un groupe peut s'évaluer à l'aide d'un critère statistique appelée "inertie intra-classe", ou "variance intra-classe". Il s'agit de la somme des distances au carré entre chaque observation et le centre ou la moyenne du groupe auquel elle appartient. nous souhaitons que l'inertie intra-classe soit la plus petite que possible.

- **Inertie inter-classe**. L'hétérogénéité entre les groupes peut se mesurer par "inertie inter-classe", ou "variance expliquée par les classes". Elle est définie par la somme des distances au carré entre chaque centre (ou moyenne) d'un groupe et le centre (ou moyenne) de l'ensemble des observations. Plus l'inertie inter-classe est grande, meilleur est le *clustering*.

Le théorème de Huygen nous fournit une base pertinente de décomposition de l'inertie totale :

$$\text{Inertie totale} = \text{Inertie intra-classe} + \text{Inertie inter-classe}$$

Or comme l'inertie totale restera constante quelle que soit la répartition, minimiser l'inertie intra-classe revient à maximiser l'inertie inter-classe et vice versa. Par conséquent, Nous pouvons mesurer la qualité de *clustering* :

- Soit par l'inertie intra-classe. Le but est alors de minimiser l'inertie intra-classe.
- Soit par l'inertie inter-classe. Le but est alors de maximiser l'inertie inter-classe.

En reposant sur ces deux critères alternatifs, en pratique, d'autres critères sont également employés. Ils ne sont rien d'autres que les variantes de ces deux critères. L'un des critères est le ratio :

$$\frac{\text{Inertie inter-classe}}{\text{Inertie totale}}$$

Il est important de souligner le fait que l'inertie inter-classe augmente et l'inertie intra-classe diminue avec le nombre de groupes. Dans le cas extrême où chaque observation forme un groupe, l'inertie inter-classe atteint son maximum et l'inertie intra-classe son minimum.

3.3.2.1 K-means

L'algorithme des K-means est l'un des algorithmes de *clustering* les plus populaires. Il s'agit d'un algorithme itératif. Il se nomme aussi l'algorithme de centres mobiles car à chaque itération, les centres sont déplacés et l'algorithme converge rapidement vers un minimum local de sa fonction objectif.

La distance par défaut étant souvent la distance euclidienne, sa fonction objectif à minimiser se définit alors comme la somme de distance euclidienne au carré

entre chaque observation et le centre auquel il est affecté :

$$\sum_{k=1}^{K} \sum_{x_j \in C_k} \left\| x_j - \mu_k \right\|^2$$

où

- K désigne le nombre de classes ;
- C_k désigne la k-ème classe ;
- x_j désigne la j-ème observation ;
- μ_k désigne le barycentre des observations au sein de la classe C_k.

Voici les différentes étapes de l'algorithme des K-means :

0. Définir *a priori* le nombre de groupes : K ;
1. Initialiser les centres des groupes de façon arbitraire. En pratique, souvent K observations sont tirées au hasard et nous les notons $\mu_1, \ldots, \mu_K$.
2. Affecter chaque observation au centre duquel elle est le plus proche

$$z_i \leftarrow argmin_k \|x_i - \mu_k\|^2$$

avec

- μ_k le k-ème centre ;
- x_i la i-ème observation ;
- z_i le numéro de classe à laquelle nous affectons x_i.

3. Mettre à jour les centres : à l'intérieur de chaque groupe, le centre est remplacé par le barycentre des observations.

$$\mu_k = \frac{1}{n_k} \sum_{z_i = k} x_i \quad k = 1, ..., K$$

où n_k est le nombre d'observations au sein du k-ème groupe et nous supposons que toutes les observations ont le même poids.

4. Répéter étape 2 et 3 jusqu'à la convergence, c'est-à-dire que les affectations des observations aux centres ne changent plus.

La Figure 3.11 donne un exemple simple de l'algorithme des K-means.

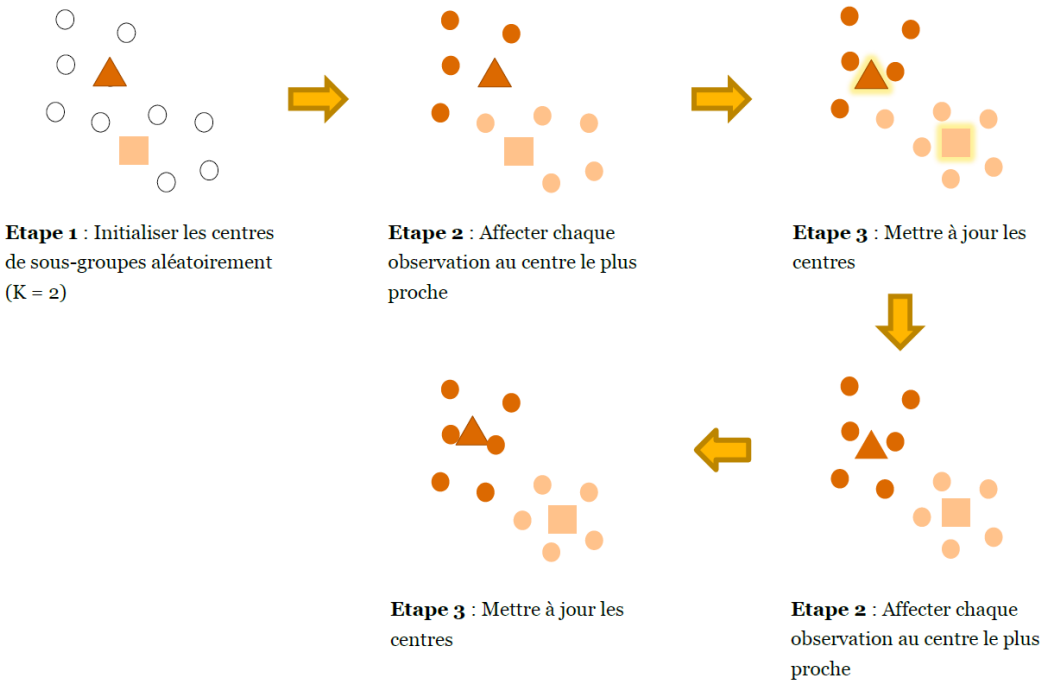

FIGURE 3.11 – Illustration de K-means

L'algorithme des K-means est très bien adapté face à un très grand jeu de données et il est beaucoup plus rapide que d'autres algorithmes de *clustering*. Mais il converge vers un optimum local et est très sensible à l'initialisation des centres. Pour y remédier , Nous pouvons avoir recours à l'algorithme de K-means ++ avec une initiation "intelligente".

Initialization avec K-means ++

1. Choisir le premier centre aléatoirement parmi toutes les observations ;
2. Pour chaque observation x, calculer la distance $d(x)$ entre x et le centre le plus proche ;
3. Choisir un nouveau centre parmi toutes les observations, avec la probabilité de choisir x proportionnelle à $d(x)^2$;
4. Répéter l'étape 2 et 3 jusqu'à obtenir K centres.

Comparé à l'initialisation aléatoire, l'initialisation de K-means ++ est plus coûteuse, mais elle permettra une convergence plus rapide de l'algorithme. De plus, grâce à K-means ++, nous avons plus de chance d'atteindre l'optimum

global. Dans les applications au Chapitre 3, K-means++ sera mis en place au lieu de K-means.

Choix du nombre de groupes

Par ailleurs, le nombre de groupes est aussi essentiel. Nous risquons d'aboutir à une partition sous-optimale avec un mauvais choix de ce nombre. Toutefois la plupart du temps nous ne serons pas en mesure de connaître le meilleur choix de ce nombre *a priori*, et il n'existe pas de règle absolue qui peut garantir le choix optimal. Donc en pratique, pour déterminer le nombre de groupes, nous effectuons plusieurs fois K-means, chaque fois avec un nombre de groupes différent ; puis nous choisissons un nombre de groupes à l'aide de plusieurs critères.

L'un des critères souvent appliqués est "le critère du coude". Il consiste à étudier le graphe de la proportion $\frac{\text{l'inertie inter-classe}}{\text{l'inertie totale}}$ en fonction du nombre de groupes ; le nombre de groupes est choisi tel que la proportion $\frac{\text{l'inertie inter-classe}}{\text{l'inertie totale}}$ ne soit pas significativement augmentée lorsque le nombre de groupes augmente. D'où l'expression de "coude". Mais ce critère ne nous donnera qu'un nombre de groupe approximatif.

Dans la littérature de classification non-supervisée, d'autres critères sont également proposés, par exemple l'indice de Davies-Bouldin et l'indice de Silhouette. Mais comme "le critère de coude", ces indices ne sont pas parfaits. En outre, à cause de la limite de la mémoire de l'ordinateur, ces indices ne sont pas toujours calculables. Dans ce mémoire, nous choisirons le nombre de groupes selon d'autres critères précisés à la section 2.2.

Il est aussi important de noter qu'avec le nombre de groupes fixé, pour obtenir une partition la plus optimale possible, nous exécutons souvent un grand nombre de fois et nous gardons la meilleure solution obtenue.

3.3.2.2 Self-Organizing Map

L'algorithme de la carte de Kohonen, ou Self-Organizing Map (SOM) , ou carte d'auto-organisation est un algorithme d'apprentissage non supervisé qui a été introduit par Teuvo Kohonen dans les années 80. Comparé à d'autres algorithmes de *clustering*, la caractéristique principale de SOM consiste réside dans le fait qu'il permet de regrouper les observations en classes tout en respectant la topologie de l'espace des observations, de garder la distance entre les observations

et de refléter les ressemblances entre les observations d'entrée. Plus précisément, les observations qui se trouvent proches entre elles dans l'espace d'entrée auront aussi des représentations proches et vice versa. En d'autres termes, nous gardons toujours le même voisinage.

L'algorithme SOM est en fait un algorithme de réseau de neurones particulier. Il est constitué de deux couches : la couche d'entrée et la couche de sortie. Le premier est l'espace d'entrée et le dernier l'espace de sortie. Les observations initiales sont présentées à la couche d'entrée. Elles sont ensuite projetées sur la couche de sortie composée d'une grille de neurones.

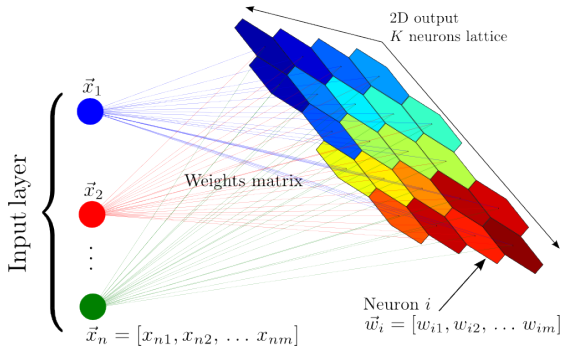

FIGURE 3.12 – Illustration d'une grille de neurones de SOM

Il s'agit d'un algorithme itératif. Plusieurs variantes de cet algorithme sont proposées dans la littérature et ici nous ne présentons qu'une version simple à titre d'illustration. On commence d'abord par introduire certains vocabulaires.

Vocabulaire

Supposons que nous disposons de N observations dans $\mathbb{R}^p$, notées $x_1, \ldots, x_i, \ldots, x_N$ et que nous les projetons sur une grille rectangulaire de K neurones. Chaque neurone a d'autres neurones comme voisins. Ainsi nous définissons une notion de voisinage. Dans certaines littératures, un neurone est aussi appelé "unité", ou "*unit*" en anglais.

À chaque neurone k $(k = 1, \ldots, K)$ nous associons un vecteur de référence, ou "vecteur code", ou "vecteur poids", ou encore "*codebook vector*". Nous désignons chaque vecteur de référence par ω_k $(k = 1, \ldots, K)$ qui est représenté par ses p composantes qui est de la même dimension que celle de l'espace des observations. Le

vecteur de référence joue le rôle de prototype, c'est-à-dire qu'il sera le représentant de chaque groupe.

Algorithme

1. Initialisation : les vecteurs de référence sont initialisés de manière aléatoire ;
2. À chaque étape, à l'itération $t+1$:
   (a) Une observation $x^{(t+1)}$ tirée au hasard est présentée ;
   (b) Elle est comparée avec tous les vecteurs de référence, c'est-à-dire que nous calculons les distances euclidiennes entre cette observation et tous les vecteurs de référence :

$$d(x^{(t+1)}, \omega_k) = \left\| x^{(t+1)} - \omega_k \right\| \quad \text{pour } k = 1, \ldots, K$$

   (c) Nous déterminons le neurone gagnant, ou "*best matching unit*" (BMU), ou "*winning unit*" : le neurone dont le vecteur de référence qui minimise cette distance "ressemble" le plus à cette observation et par conséquent il sera le neurone gagnant. Si nous désignons ce neurone gagnant par $k^{(t+1)\bigstar}$, alors Nous pouvons ainsi formaliser :

$$k^{(t+1)\bigstar} = argmin_{k=1}^{K} \left\{ \left\| x^{(t+1)} - \omega_k^{(t)} \right\| \right\}$$

   L'observation $x^{(t+1)}$ est alors cartographiée à ce neurone gagnant.
   (d) Ensuite, nous rapprochons ce neurone gagnant en mettant à jour son vecteur de référence. Nous rapprochons aussi les neurones voisins en mettant à jour les vecteurs de référence associés à ces neurones de la même façon :

$$\omega_k^{(t+1)} = \omega_k^{(t)} + h_{k^{(t+1)}\bigstar k}^{(t+1)} \times (x^{(t+1)} - \omega_k^{(t)})$$

   où $h_{k^{(t+1)}\bigstar k}^{(t+1)}$ appelée la fonction de voisinage, se définit comme suivante :

$$h_{k^{(t+1)}\bigstar k}^{(t+1)} = \alpha(t+1) \cdot \exp\left\{ - \frac{\left\| \omega_{k^{(t+1)}\bigstar}^{(t+1)} - \omega_k^{(t)} \right\|^2}{2\sigma^2(t+1)} \right\}$$

   où α et σ sont respectivement deux fonctions réelles décroissantes de t. Nous ne détaillons pas explicitement les formes mathématiques de α

ou σ car différentes variantes de SOM proposent différentes formes de ces deux fonctions. Il existe aussi d'autres formes mathématiques de $h^{(t+1)}_{k^{(t+1)\star}k}$. L'une des plus simples est

$$h^{(t+1)}_{k^{(t+1)\star}k} = \begin{cases} 1, & \text{si } \left\| \omega^{(t+1)}_{k^{(t+1)\star}} - \omega^{(t)}_{k} \right\|^2 \leq \delta \\ 0, & \text{sinon} \end{cases}$$

avec δ étant un seuil pré-défini. Cette fonction de voisinage s'interprète comme suivante : les vecteurs de référence sont mis à jour lorsqu'ils sont assez proche de celui du neurone gagnant.

Ainsi, les neurones voisins auront les vecteurs de référence similaires et la topologie est conservée. C'est pour cette raison-là que l'algorithme est nommé self-organizing map, ou carte d'auto-organisation en français.

Enfin, à l'issue de l'apprentissage, chaque observation appartient à un neurone qui est le neurone gagnant lorsque elle est présentée. De ce fait, nous pouvons voir les neurones comme des classes et donc chaque observation appartient à une classe qui est le neurone auquel elle appartient. Les vecteurs de référence associés sont comme des prototypes de chaque classe. De plus, les observations qui sont proches dans l'espace d'origine sont cartographiées dans le même neurone ou dans les neurones voisins, et ainsi nous obtenons une carte qui conserve bien la topologie.

En un sens, l'algorithme SOM peut être vu comme une généralisation de l'algorithme des K-means (Ripley 1996). Dans ce sens, chaque vecteur de référence correspond à un centre local d'un groupe de K-means et le nombre de groupes est déterminé *a priori* par la taille de la grille. La différence principale par rapport aux K-means consiste au fait que SOM permet aussi de conserver la topologie de l'espace d'origine et la carte de neurones garde la distance entre les observations d'entrée.

Deux-étape clustering

La taille de grille est un choix arbitraire et il n'existe pas une méthode pour choisir la taille de grille de manière optimale. D'après l'oeuvre originale de Kohonen, la taille de grille est déterminée via la méthode essai-erreur : nous effectuons

des essais plusieurs fois jusqu'à obtention du résultat satisfaisant. Toutefois, heuristiquement, afin d'obtenir une segmentation assez satisfaisante, de nombreux auteurs proposent d'utiliser une grille "relativement grande". Toutefois, il n'est pas facile de donner d'interprétation ou extraire de l'information utile en présence d'une grande grille. De plus, nous attendons souvent que le nombre de classes soit moindre que celui issu d'une grande grille. Par conséquent, en pratique, le clustering se fait en deux étapes :

- Dans un premier temps, nous nous donnons souvent une taille de grille assez grande, par exemple 10×10, i.e. 100 neurones ou classes. À l'issue de l'algorithme SOM, nous obtenons une grille de neurones, où chaque neurone est associé à un vecteur de référence qui représente la classe.
- Dans un second temps, à partir de cette grille de neurones, nous effectuons un deuxième regroupement sur les vecteurs de référence via les K-means ou la classification hiérarchique.

Ainsi nous définissons un *clustering* de deux niveaux : le premier étant les classes issues de SOM, appelé "micro-classes" et le deuxième étant les classes issues des K-means ou de la classification hiérarchique, appelé "macro-classes" ou "super-classes".

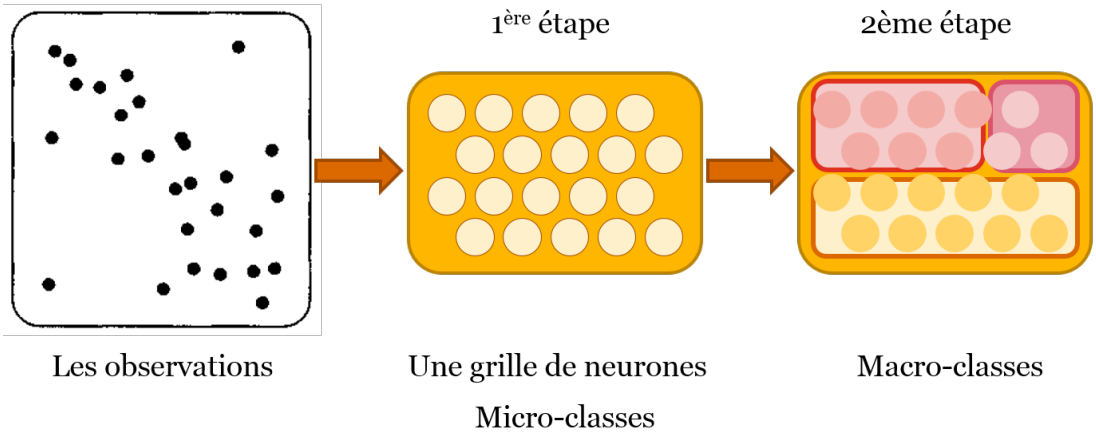

FIGURE 3.13 – Illustration de deux-étape clustering

3.3.2.3 Modèle de mélange gaussien

Rappelons la définition du modèle de mélange de lois introduit au Chapitre 1. Supposons qu'il y ait un nombre fini de groupes. Au sein de chaque groupe,

les observations suivent une même loi de probabilité. En mélangeant tous ces groupes, nous obtenons un ensemble d'observations. À présent, si nous tirons une observation x de cet ensemble, sa densité vaut alors :

$$f(x|\theta) = \sum_{k=1}^{K} \pi_k f_k(x|\theta_k) \tag{3.3}$$

où

— K désigne le nombre de groupes ;

— Les π_k décrivent la proportion des observations appartenant au groupe k avec $0 \leq \pi_k \leq 1$ et $\sum_{k=1}^{K} \pi_k = 1$;

— $f_k(.|\theta_k)$ correspond à la densité associée au groupe k avec les paramètres θ_k.

Un exemple typique de modèle de mélange de lois est le modèle de mélange gaussien, où toutes les $f_k(.|\theta_k)$ dans la formule (2.3) sont des lois gaussiennes. Il est usuellement abrégé par l'acronyme anglais GMM pour Gaussian Mixture Model.

Comme nous avons déjà mentionné au Chapitre 1 que l'hétérogénéité des risques peut se formaliser mathématiquement par les modèles de mélange de lois, il est alors naturel de penser à effectuer une classification non-supervisée via un modèle de mélange de lois. Il convient cependant de remarquer que l'estimation des paramètres de chaque loi composante reste toujours un sujet très difficile à traiter. Par conséquent en pratique seuls les mélanges de lois gaussiennes sont considérés. Sous hypothèse de mélanges de lois gaussiennes, au sein de chaque groupe k, les observations suivent une loi gaussienne de moyenne μ_k et de matrice de covariance Σ_k. Donc chaque groupe est caractérisé par le tuple (π_k, μ_k, Σ_k) .

Estimation des paramètres : l'algorithme EM

L'algorithme EM, i.e. Espérance-Maximisation est introduit afin d'estimer les paramètres d'un modèle de mélange de lois. Tout comme son nom l'indique, le principe de cet algorithme consiste à itérer les deux étapes suivantes jusqu'à la condition d'arrêt :

1. **Étape d'évaluation de l'espérance (E)** : étant donné les paramètres estimés du modèle $\{(\hat{\pi_k}, \hat{\mu_k}, \hat{\Sigma}_k)\ k = 1, ..., K\}$, nous calculons la probabilité que l'observation i soit affectée au k-ième groupe. nous notons cette probabilité

r_{ik}. Par le théorème de Bayes, nous obtenons

$$\begin{aligned} r_{ik} &= \mathbb{P}\Big(x_i \in C_k | \{(\hat{\pi_k}, \hat{\mu_k}, \hat{\Sigma_k}) \ k = 1, ..., K\}, x_i\Big) \\ &= \frac{\mathbb{P}\Big(x_i \in C_k | \{(\hat{\pi_k}, \hat{\mu_k}, \hat{\Sigma_k}) \ k = 1, ..., K\}\Big) \cdot f\Big(x_i | x_i \in C_k, \{(\hat{\pi_k}, \hat{\mu_k}, \hat{\Sigma_k}) \ k = 1, ..., K\}\Big)}{f\Big(x_i | \{(\hat{\pi_k}, \hat{\mu_k}, \hat{\Sigma_k}) \ k = 1, ..., K\}\Big)} \end{aligned}$$

où C_k désigne le k-ème groupe.
Or

$$\begin{aligned} \mathbb{P}\Big(x_i \in C_k | \{(\hat{\pi_k}, \hat{\mu_k}, \hat{\Sigma_k}) \ k = 1, ..., K\}\Big) &= \hat{\pi}_k \\ f\Big(x_i | x_i \in C_k, \{(\hat{\pi_k}, \hat{\mu_k}, \hat{\Sigma_k}) \ k = 1, ..., K\}\Big) &= f^{\mathcal{N}}(x_i | \mu_k, \Sigma_k) \end{aligned}$$

où $f^{\mathcal{N}}(.|\mu_k, \Sigma_k)$ désigne la densité d'une loi normale de moyenne μ_k et de matrice de covariance Σ_k.

Donc Nous pouvons réécrire

$$r_{ik} = \frac{\hat{\pi}_k f^{\mathcal{N}}(x_i | \hat{\mu_k}, \hat{\Sigma_k})}{\sum_{j=1}^{K} \hat{\pi}_j f^{\mathcal{N}}(x_i | \hat{\mu_j}, \hat{\Sigma_j})} \tag{3.4}$$

Ainsi, pour l'observation i, nous pouvons définir un vecteur

$$r_i = \begin{bmatrix} r_{i1} \\ ... \\ r_{ik} \\ ... \\ r_{iK} \end{bmatrix}$$

2. **Étape de maximisation (M)** : étant donné l'appartenance des groupes pour toutes les observations $\hat{r}_{ik}, x_i$, Nous pouvons alors estimer les paramètres (π_k, μ_k, Σ_k) $k = 1, ...K$ via le maximum de vraisemblance.

La Figure 3.14 montre une application d'un modèle de mélanges gaussiens dans une étude du temps d'attente en minutes avant la prochaine éruption d'un

geyser [3]. L'histogramme de la variable à étudier "temps d'attente" est de couleur grise. Les deux densités gaussiennes identifiées par le GMM via l'algorithme EM sont en couleur orange et fuchsia respectivement.

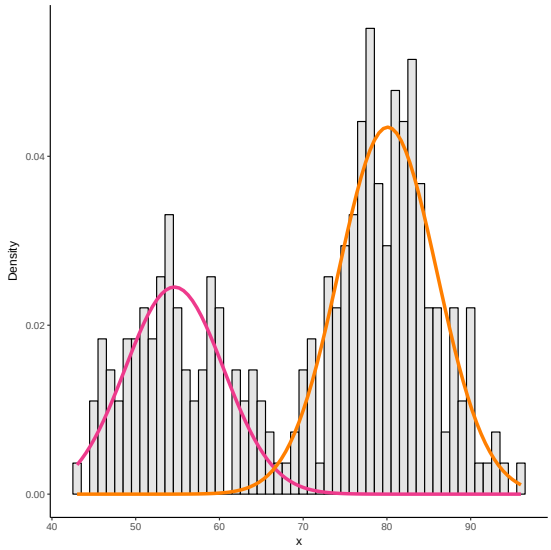

FIGURE 3.14

3. Il s'agit de la base de données *Faithful* dans le logiciel R. Cette base de données décrit les éruptions d'un geyser situé dans le parc national de yellowstone aux États-Unis.

Chapitre 4

Mise en application

Après avoir, dans le chapitre précédent, expliqué les théories des algorithmes, il est maintenant temps de mettre en application les différentes approches et méthodes. Nous commençons par l'étude statistique descriptive des données dans la première section. Les différentes approches et méthodes sont ensuite présentées dans la deuxième section. Nous terminerons ce chapitre sur la comparaison des résultats.

4.1 Étude statistique descriptive des données

La base de données qui sert de support à notre étude est un portefeuille d'assurance automobile. Pour des raisons de confidentialité, les données ont été anonymisées mais la structure des données n'est pas modifiée. La base d'étude est la base de contrats. Elle comporte 180 000 lignes, chaque ligne correspond à un individu qui a eu des sinistres. Elle a 131 colonnes : 130 variables explicatives qui donnent l'information sur les caractéristiques des assurés et une variable à prédire – le montant des sinistres. Parmi les 130 variables explicatives, il y a 100 variables qualitatives et 30 variables quantitatives.

cat2	cat3	cat4	cat5	cat6	cat7	cat8	cat9	cat10	...	cont6	cont7	cont8	cont9	cont10	cont11	cont12	cont13	cont14	loss
B	A	B	A	A	A	A	B	A	...	0.718367	0.335060	0.30260	0.67135	0.83510	0.569745	0.594646	0.822493	0.714843	2213.18
B	A	A	A	A	A	A	B	B	...	0.438917	0.436585	0.60087	0.35127	0.43919	0.338312	0.366307	0.611431	0.304496	1283.60
B	A	A	B	A	A	A	B	B	...	0.289648	0.315545	0.27320	0.26076	0.32446	0.381398	0.373424	0.195709	0.774425	3005.09
B	A	B	A	A	A	A	B	A	...	0.440945	0.391128	0.31796	0.32128	0.44467	0.327915	0.321570	0.605077	0.602642	939.85
B	A	B	A	A	A	A	B	B	...	0.178193	0.247408	0.24564	0.22089	0.21230	0.204687	0.202213	0.246011	0.432606	2763.85

FIGURE 4.1 – Illustration de la base de donnée étudiée

Étudions la variable à prédire : le montant des sinistres.

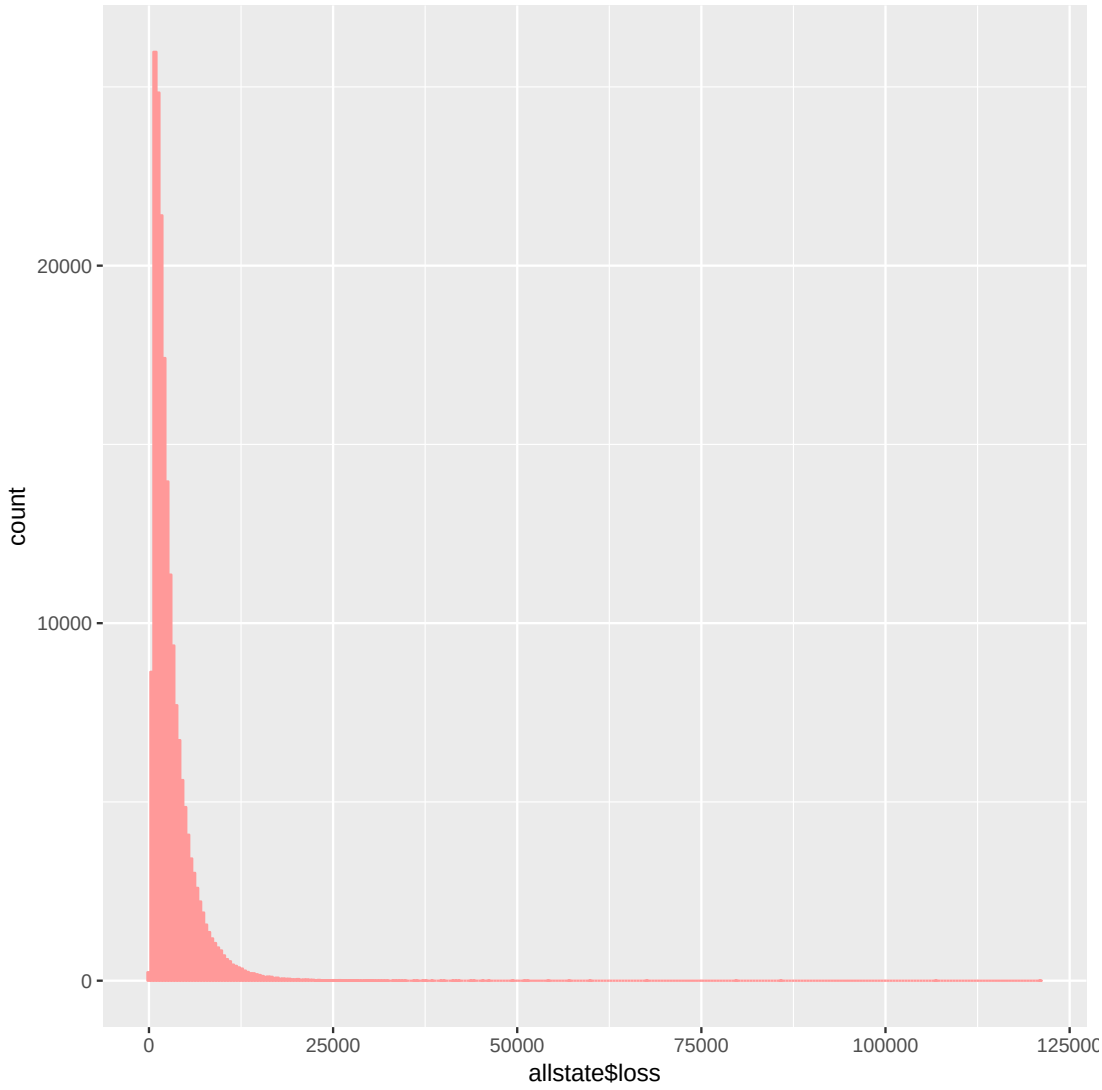

FIGURE 4.2 – L'histogramme de la distribution de montant des sinistres

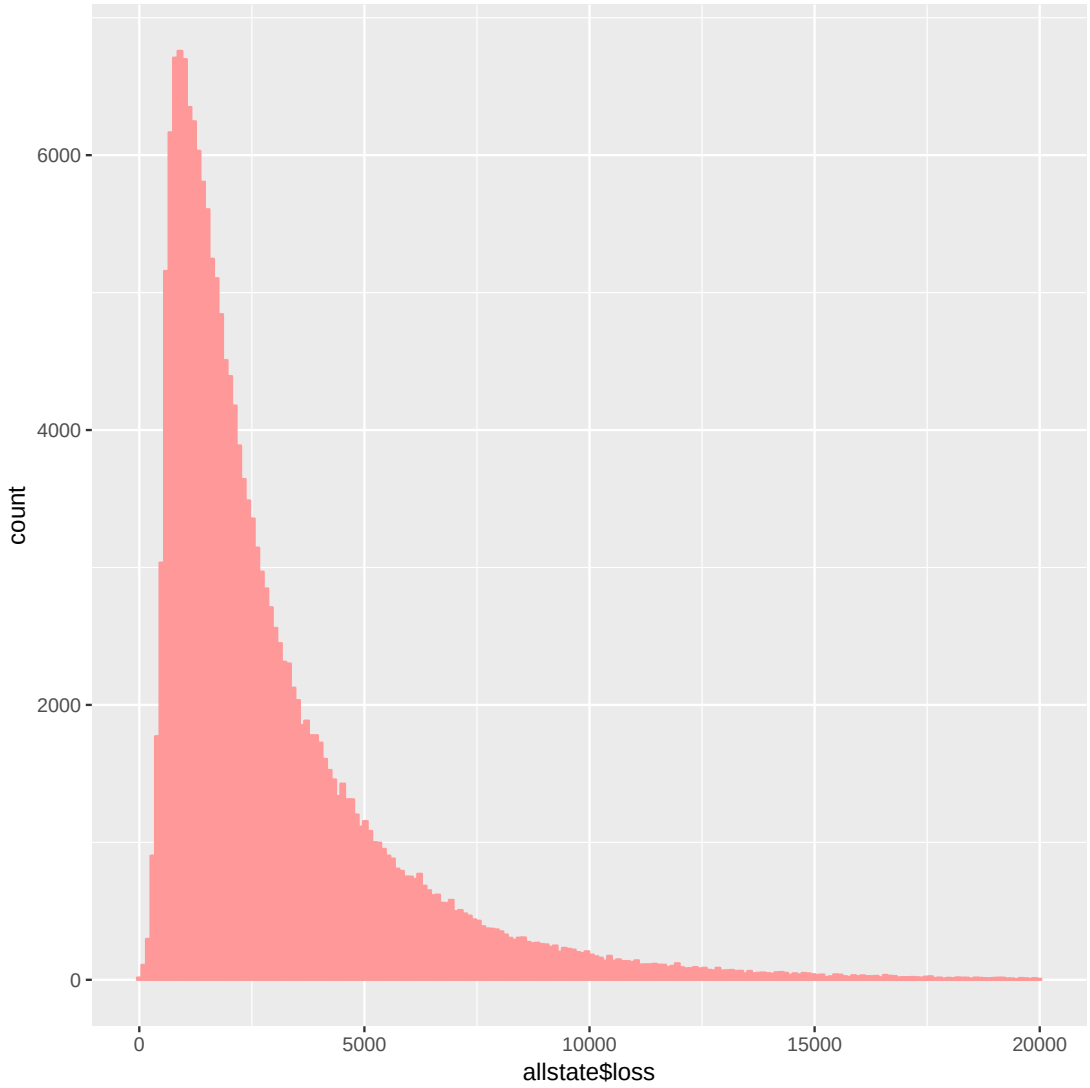

FIGURE 4.3 – Zoom sur les montants de sinistres inférieurs à 20000

La Figure 4.2 et la Figure 4.3 nous montrent l'histogramme de la distribution des montants des sinistres. La plupart des montants des sinistres est inférieurs à 3864 (environ 75% des sinistres dans la base), avec toutefois des sinistres extrêmes qui peuvent atteindre 121 012.

Les caractéristiques de cette variable sont présentées dans le tableau ci-dessous (Figure 4.4).

Minimum	1er quartile	Médiane	Moyenne	3ème quartile	Maximum
0,67	1204,46	2115,57	3037,34	3864,05	121 012,25

FIGURE 4.4

4.2 Description des approches et méthodes

Afin d'avoir un cadre cohérent de test de la pertinence de partitionnement en groupes tout au long de ce chapitre, une version simple de la méthode de validation croisée (*cross validation*) sera appliquée. Pour chaque méthode, nous séparons la base de données en deux sous-échantillons : l'échantillon d'apprentissage et l'échantillon de test. Le première contient 75% des données et la dernière contient 25%. Toutes les observations dans l'échantillon d'apprentissage sont tirés de façon aléatoire. Toutefois, l'échantillon d'apprentissage et l'échantillon de test que nous fournissons à chaque méthode restent les mêmes, de sorte que nous puissions faire des comparaisons sur le même échantillon de test à la section suivante.

Rappelons les deux approches présentées à la section 2.3 : l'approche *a posteriori* et l'approche *a priori*. Nous commençons par détailler les différentes méthodes dans l'approche *a posteriori*.

4.2.1 L'approche *a posteriori*

4.2.1.1 1ère méthode : GMM sur les résidus de déviance

Algorithme

1. **Séparation des échantillons d'apprentissage et de test**. L'ensemble d'échantillon est d'abord séparé en l'échantillon d'apprentissage et l'échantillon de test. Sur l'échantillon de test, nous cachons la variable à prédire "montant des sinistres". Les observations au sein de l'échantillon de test peuvent être considérées comme les "nouveaux arrivants". L'assureur dispose d'informations sur leurs caractéristiques : âge, éducation, profession, etc. Mais leurs montants des sinistres sont inconnus.

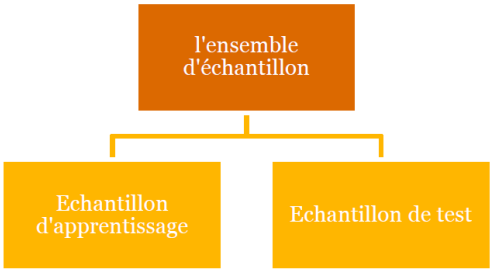

FIGURE 4.5 – L'étape 1 : Séparation des échantillons d'apprentissage et de test

2. **Calibrage du modèle de régression via XGBoost.** Nous calibrons un modèle de régression d'XGBoost sur l'ensemble de l'échantillon d'apprentissage. Ce modèle prend les caractéristiques des assurés notées X comme entrée et il permet de prédire les montants des sinistres.

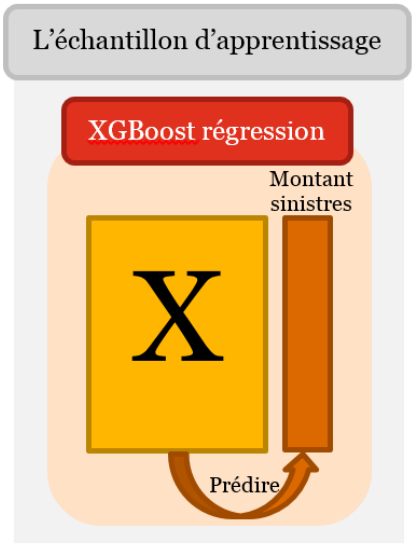

FIGURE 4.6 – L'étape 2 : Calibrage du modèle de régression via XGBoost

3. **Calcul des résidus.** Toujours sur l'échantillon d'apprentissage, le modèle construit est utilisé pour prédire les montants des sinistres puis les résidus de déviance sont en déduits.

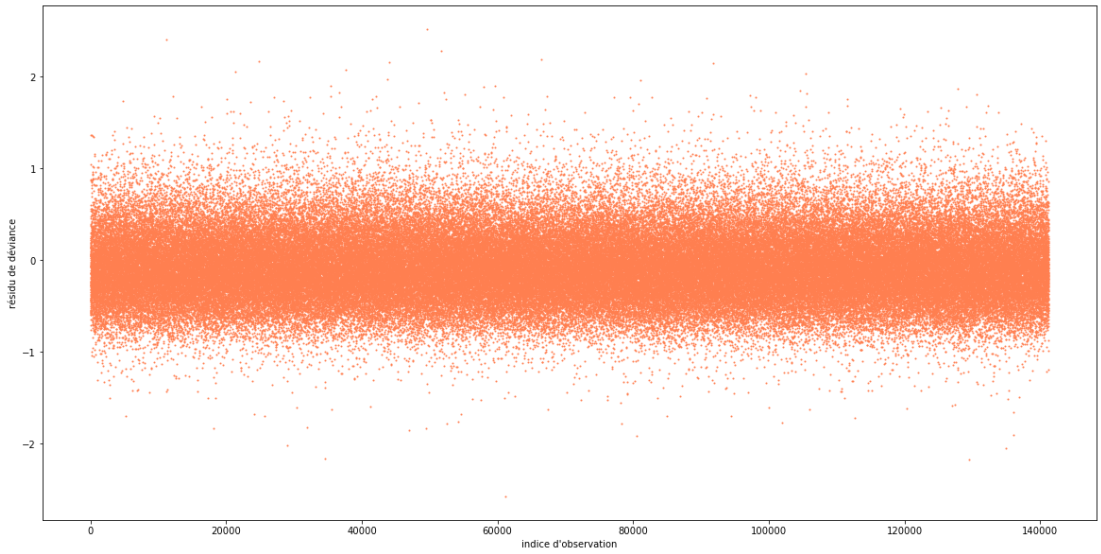

FIGURE 4.7 – Index plot

La Figure 4.7 présente les indices d'observations en abscisse et les résidus de déviance en ordonnée. Ce graphique nous permet d'identifier les observations ayant des résidus de déviance trop élevés. Nous constatons que la plupart des résidus se situent autour de zéro, ce qui signifie que globalement le modèle calibré donne une bonne prédiction.

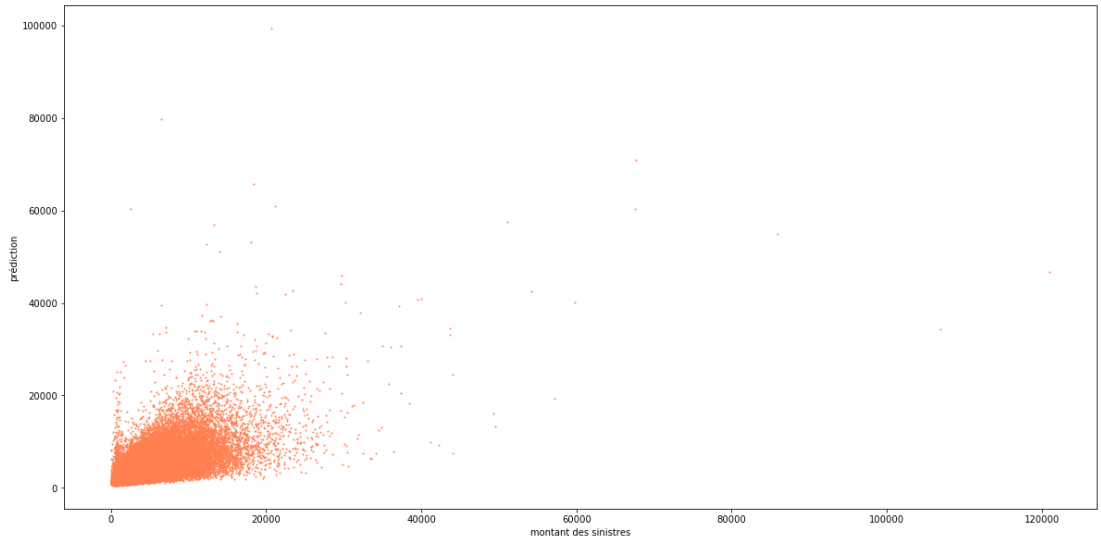

FIGURE 4.8 – Précisions versus les vraies valeurs des montants des sinistres

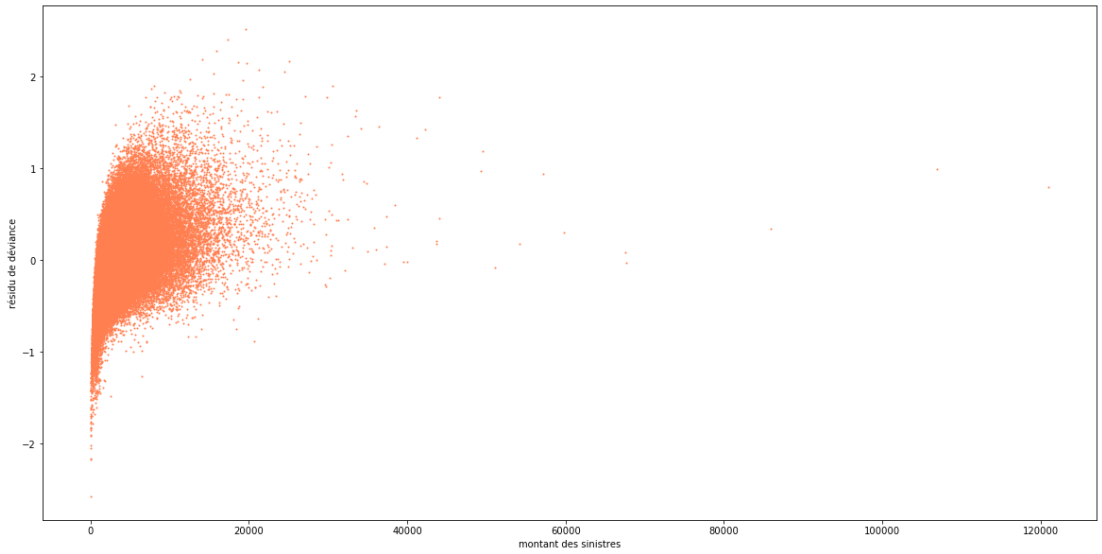

FIGURE 4.9 – Résidus versus les vraies valeurs des montants des sinistres

Toutefois, lorsque nous présentons les prédictions des montants des sinistres en fonction des vraies valeurs (Figure 4.8), nous constatons qu'il existe une tendance que le modèle n'a pas réussi à capturer. Cette tendance indique aussi que l'hypothèse sur la distribution n'est pas exacte. Cela peut s'expliquer par l'hétérogénéité des risques. Il en est de même lorsque nous présentons les résidus de déviance en fonction des montants de sinistres (Figure 4.9). Nous espérons que nous pouvons améliorer le modèle dans les étapes suivantes.

4. **Classification des résidus via GMM.** Comme les résidus doivent être distribués normalement approximativement [1], nous effectuons une classification non-supervisée sur les résidus via le modèle de mélanges gaussiens (GMM). Il s'agit du GMM implémenté dans le package *sklearn* du Python. Il donne directement l'appartenance des groupes. Nous testerons le nombre de groupes variant de 2 à 10 pour comparer les performances des modèles.

1. Dan Tevet, *And the winner is... ? How to pick a better model : Part 2 Goodness-of-fit and internal stability*, Verisk Insurance Solutions, P12-13.

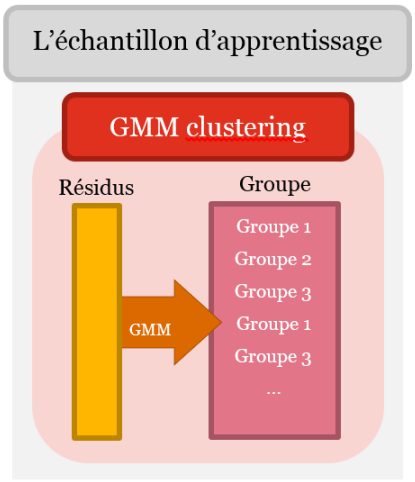

FIGURE 4.10 – L'étape 4 : Classification des résidus via GMM

5. **Calibrage du modèle de classification via XGBoost.** Étant donné l'appartenance aux groupes et X, un modèle de classification via XGBoost est ensuite calibré sur l'échantillon d'apprentissage tel que ce modèle puisse prédire les probabilités d'appartenance de telle ou telle observation à quel groupe. Cette étape est nécessaire car pour un nouveau arrivant ou une observation au sein de l'échantillon de test, son montant des sinistres est inconnu. Donc nous ne connaissons pas son résidu de déviance ni son appartenance aux groupes.

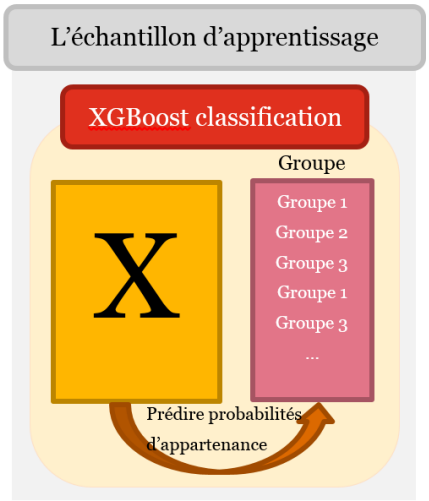

FIGURE 4.11 – L'étape 5 : Calibrage du modèle de classification via XGBoost

6. **Prédiction par le modèle de classification via XGBoost.** [2] Une fois le modèle de classification calibré, il est utilisé pour prédire les probabilités d'appartenance aux groupes sur l'échantillon de test. Plus précisément, chaque fois que nous fournissons le modèle des caractéristiques x_i d'un individu i comme entrée, le modèle donnera un vecteur de probabilités dont la 1ère coordonnée est la probabilité que x_i appartienne au groupe 1, la 2ème coordonnée est la probabilité que x_i appartienne au groupe 2, ainsi de suite.

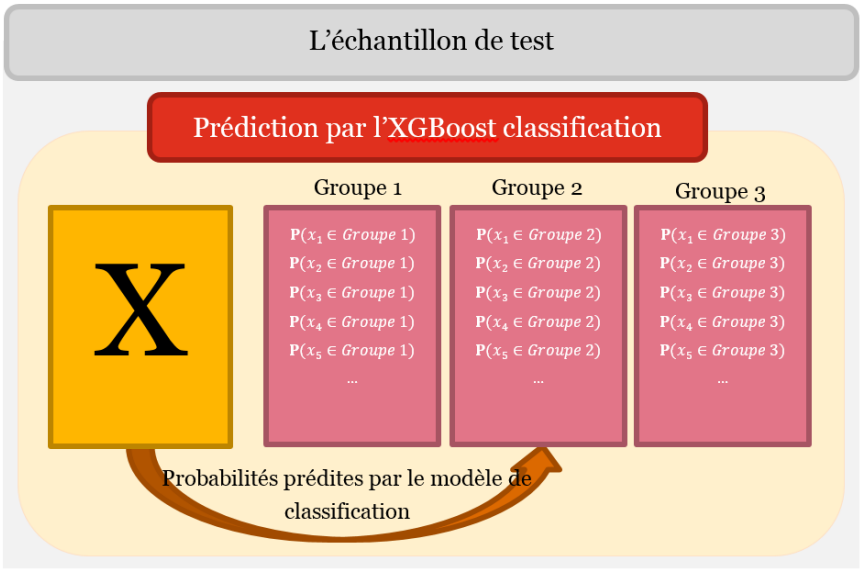

FIGURE 4.12 – L'étape 6 : Prédiction par le modèle de classification via XGBoost

7. **Calibrage d'une régression groupe par groupe via XGBoost.** Revenons sur l'échantillon d'apprentissage à présent. Les modèles de régression d'XGBoost sont calibrés respectivement sur chaque groupe. Par exemple, pour le groupe k, l'XGBoost va prendre les caractéristiques des assurés X au sein du groupe k comme entrée, et essayer de prédire les montants des sinistres. L'XGBoost construit sur le groupe k est noté XGB_k.

2. Nous aurions pu aussi utiliser une simple classification bayésienne. Si le choix est porté sur la classification via XGBoost, c'est parce que d'une manière générale l'XGBoost réalise de meilleures performances et il est également plus rapide.

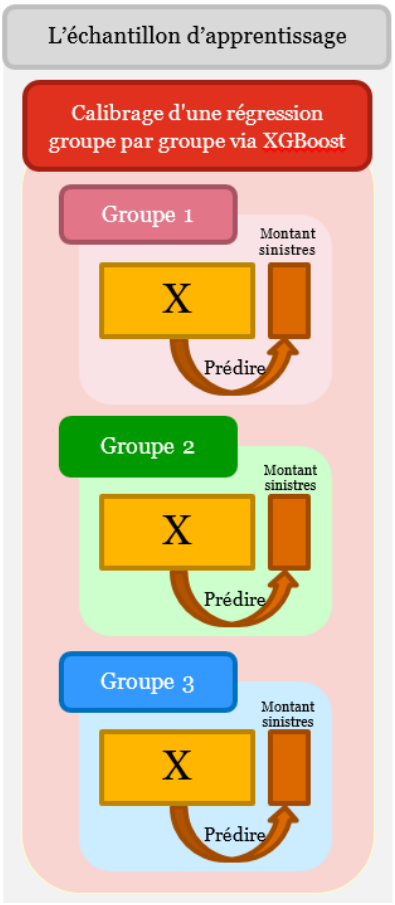

FIGURE 4.13 – L'étape 7 : Calibrage d'une régression groupe par groupe via XGBoost

8. **Prédiction sur l'échantillon de test.** Finalement, sur l'échantillon de test, avec une observation x_j donnée, les modèles d'XGBoost calibrés dans l'étape précédente sont utilisés pour faire des prédictions :

   — Si nous supposons que $x_j \in$ groupe 1, la prédiction sera $\mathrm{XGB}_1(x_j)$;

   — Si nous supposons que $x_j \in$ groupe 2, la prédiction sera $\mathrm{XGB}_2(x_j)$;
   ...

   — Si nous supposons que $x_j \in$ groupe k, la prédiction sera $\mathrm{XGB}_k(x_j)$;
   ...

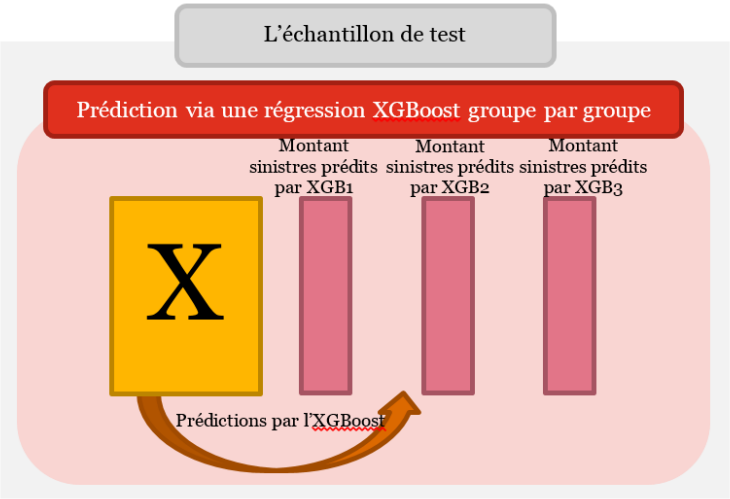

FIGURE 4.14 – L'étape 8 : Prédiction sur l'échantillon de test

Supposons qu'au total il y a K groupes et désignons les probabilités d'appartenance aux groupes obtenues via l'XGBoost classification à l'étape 6 par $\hat{\mathbf{P}}(x_j \in \text{groupe } 1), \ldots, \hat{\mathbf{P}}(x_j \in \text{groupe } k), \ldots, \hat{\mathbf{P}}(x_j \in \text{groupe } K)$, alors la prédiction du montant des sinistres pour l'observation x_j vaudra :

$$\hat{y}_j = \sum_{k=1}^{K} \hat{\mathbf{P}}(x_j \in \text{groupe } k) \cdot \text{XGB}_k(x_j).$$

La Figure 4.15 et la Figure 4.16 résument l'algorithme de la première méthode.

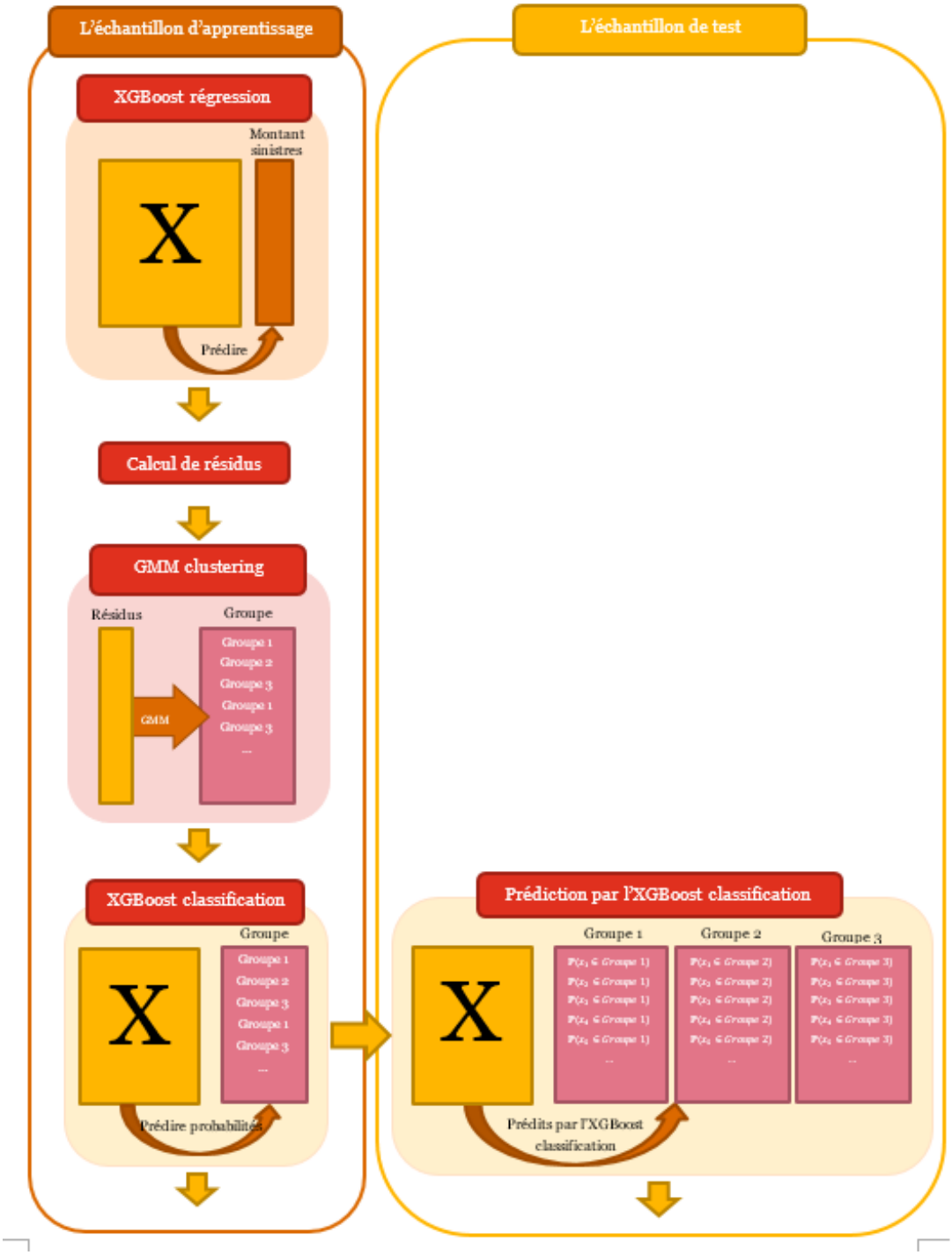

FIGURE 4.15 – Illustration de la méthode GMM sur les résidus de déviance 1

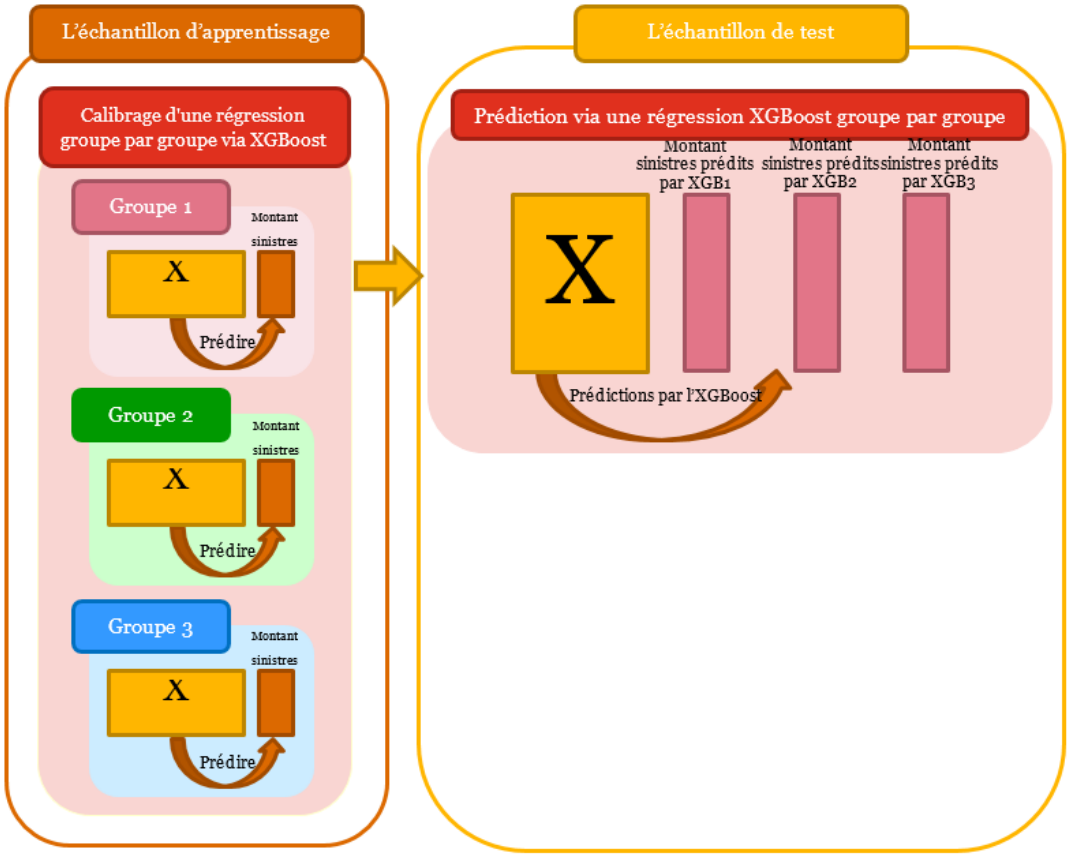

FIGURE 4.16 – Illustration de la méthode GMM sur les résidus de déviance 2

4.2.1.2 2ème méthode : K-means sur les résidus de déviance

La deuxième méthode est une variante de la première méthode. Dans cette méthode, au lieu d'effectuer une classification non-supervisée sur les résidus via GMM à l'étape 4, nous appliquerons l'algorithme des K-means. Le reste de l'algorithme reste semblable à celui de la première. Par conséquent nous ne détaillerons pas ici.

4.2.2 L'approche *a priori*

Considérons maintenant les méthodes dans l'approche *a priori*.

4.2.2.1 1ère méthode : ACP - K-means

Algorithme

1. **Réduction de la dimension via l'ACP.** D'emblée, une analyse en composantes principales est effectuée sur l'ensemble de la base de données pour réduire la dimension. Il convient de remarquer que l'ACP est uniquement appliquée sur la matrice des variables explicatives X. Il est souhaitable de conserver autant d'inertie que possible. Si nous conservons 90% d'inertie, nous obtenons 82 axes principaux. Le nombre de dimensions reste toujours assez élevé. Toutefois, il n'est pas non plus désirable de réduire la dimension au prix de la perte de trop d'information. Finalement, nous avons décidé de conserver au moins 85% d'inertie, ce qui résulte en 73 axes principaux. La nouvelle matrice de variables explicatives suite à l'ACP est notée Z (la Figure 4.17) .

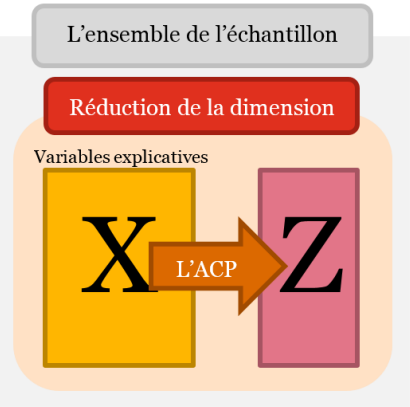

FIGURE 4.17 – L'étape 1 : Réduction de la dimension via l'ACP

2. **Séparation des échantillons d'apprentissage et test.** La base de données après la réduction de la dimension est séparée en l'échantillon d'apprentissage et l'échantillon de test (Figure 4.18). Nous désignons
   — $Z_{training}$ la nouvelle matrice des variables explicatives sur l'échantillon d'apprentissage ;

— Z_{test} la nouvelle matrice des variables explicatives sur l'échantillon de test ;

— $y_{training}$ la variable à prédire sur l'échantillon d'apprentissage ;

— y_{test} la variable à prédire sur l'échantillon de test ;

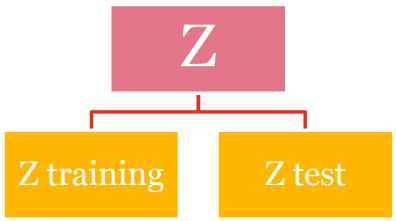

FIGURE 4.18 – L'étape 2 : Séparation des échantillons d'apprentissage et test

3. **Classification de $Z_{training}$ via K-means.** Une classification non-supervisée sur $Z_{training}$ via K-means est effectuée afin d'obtenir les groupes homogènes (Figure 4.19). Nous testerons le nombre de groupes variant de 2 à 10 pour comparer les performances des modèles.

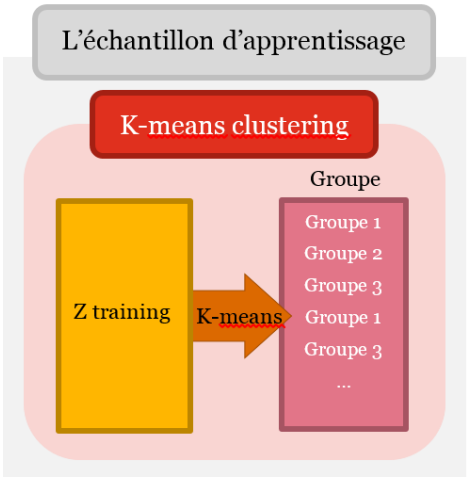

FIGURE 4.19 – L'étape 3 : Classification de $Z_{training}$ via K-means

4. **Prédiction de l'appartenance aux groupes.** Le K-means calibré permet de prédire l'appartenance aux groupes sur l'échantillon de test Z_{test} (Figure 4.20).

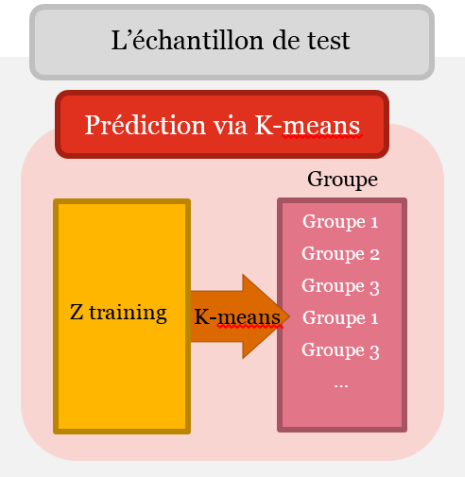

FIGURE 4.20 – L'étape 4 : Prédiction de l'appartenance aux groupes

5. **Calibrage d'une régression groupe par groupe via XGBoost.** Revenons sur l'échantillon d'apprentissage à présent. Les modèles de régression d'XGBoost sont calibrés respectivement sur chaque groupe. L'XGBoost prend $X_{training}$ par groupe comme entrée et $y_{training}$ par groupe comme la variable à prédire. L'XGBoost construit sur le groupe k se note XGB_k.
6. **Prédiction sur l'échantillon de test.** Finalement, sur l'échantillon de test, avec une observation x_j donnée, les modèles d'XGBoost calibrés dans l'étape précédente sont servis pour faire des prédictions :
   si $x_j \in$ Groupe k selon la prédiction de K-means à l'étape 4, sa prédiction de montants des sinistres vaudra :

$$\hat{y}_j = \text{XGB}_k(x_j).$$

4.2.2.2 2ème méthode : Auto-encodeur - K-means

La deuxième méthode est une variante de la première. Dans cette méthode, à l'étape 1, pour réduire la dimension, au lieu de l'ACP, l'auto-encodeur est mis en place. Il y a de nombreuses variantes de l'auto-encodeur. Ici, nous avons mis en application une version simple de l'auto-encodeur, avec une couche cachée comprenant deux neurones. Nous avons choisi le nombre de neurones égal à deux pour les raisons suivantes :

— Selon le résultat de l'auto-encodeur, avec une erreur quadratique moyenne

égale à 0.019, le résultat est déjà assez satisfaisant même avec une couche cachée de deux neurones ;

— Avoir deux neurones dans la couchée cachée facilite aussi la visualisation. Nous pouvons ainsi réduire la dimension à 2 et visualiser les données transformées. Sur la Figure 4.21 Réduction de la dimension via l'auto-encodeur, nous pouvons constater l'existence des groupes.

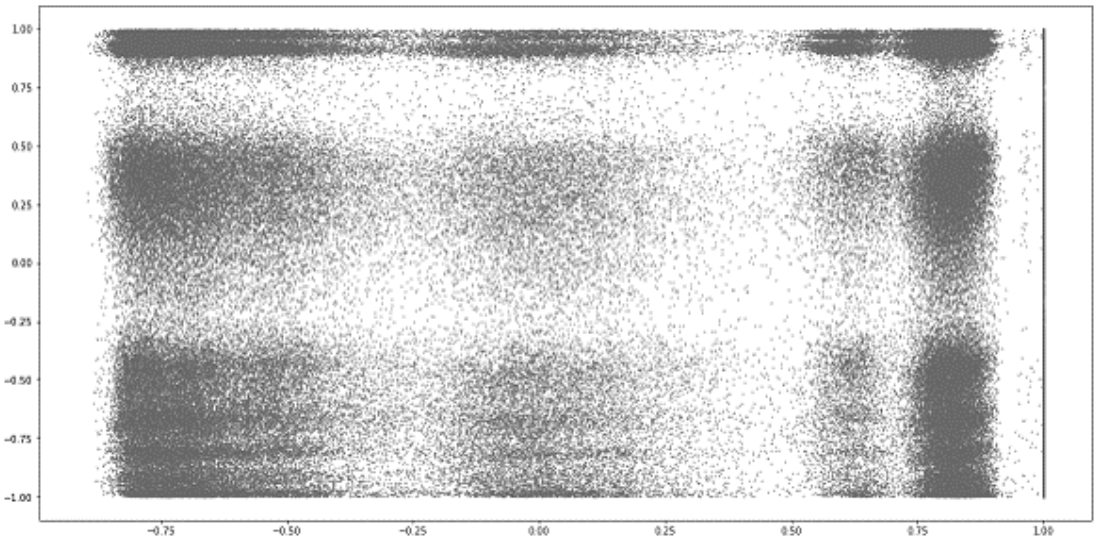

FIGURE 4.21 – Réduction de la dimension via l'auto-encodeur

Le reste de l'algorithme reste similaire à la première méthode. Nous ne détaillerons plus ici.

4.2.2.3 3ème méthode : ACP - SOM

Cette méthode est une variante de la première. De même que cette dernière, à l'étape 1, la même ACP est effectuée pour réduire la dimension de 130 à 73. Mais à l'étape 3, au lieu des K-means, le SOM est mis en place. Nous testerons le nombre de groupes variant de 2 à 10 pour comparer les performances des modèles. Le reste de l'algorithme reste pareil que la première méthode. Nous ne développerons pas plus ici.

4.2.2.4 4ème méthode : ACP - SOM - K-means

Cette méthode est une variante de la méthode précédente. Rappelons le "deux-étape clustering" décrit à la fin de la sous-sous-section 3.3.2.2 Self-Organizing Map, où dans un premier temps nous mettons en oeuvre le SOM avec une taille de grille relativement grande ; puis dans un second temps nous effectuons un deuxième regroupement sur les vecteurs de référence obtenus par les K-means. Nous allons mettre en application ce "deux-étape clustering" ici.

Après la réduction de la dimension et la séparation de training set et test set, nous allons d'abord appliquer le SOM avec une grille de taille 50×50, i.e. 2500 neurones. Puis nous effectuons l'algorithme des K-means sur les 2500 vecteurs de référence obtenus. Nous testerons le nombre de groupes de K-means variant de 2 à 10 pour comparer les performances des modèles. Le reste de l'algorithme étant pareil que la première méthode, nous ne détaillerons pas plus ici.

4.2.2.5 5ème méthode : CART sur les résidus obtenus par le modèle naïf

Algorithme

1. **Séparation des échantillons d'apprentissage et test.** Nous séparons d'abord l'ensemble d'échantillon en échantillon d'apprentissage et en échantillon de test.

2. **Définition du modèle naïf.** Un modèle naïf est défini comme suivant : les prédictions de ce modèle seront toutes pareilles : elles seront égales à la moyenne des montants des sinistres dans l'échantillon d'apprentissage.

3. **Calcul des résidus.** Toujours sur l'échantillon d'apprentissage, les résidus de déviance sont calculés à partir de ledit modèle naïf.

4. **Calibrage du modèle de régression via CART.** Comme le modèle est très naïf, nous nous attendons à ce qu'il ne prédise pas très bien et nous espérons repérer certains "patterns" dans les résidus de déviance. Un modèle de CART est calibré sur l'échantillon d'apprentissage, qui prend les caractéristiques des assurés X comme entrée et les résidus obtenus à l'étape précédente comme la valeur à prédire (la Figure 4.22)

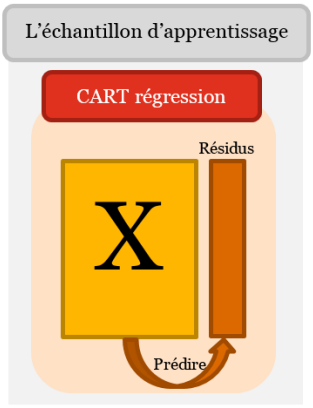

FIGURE 4.22 – Étape 4 : Calibrage du modèle de régression via CART

Comme nous l'avons mentionné à la sous-section 3.2.1 CART, à l'issue de CART, les observations au sein d'une même feuille peuvent être vues comme un groupe homogène. La figure Figure 4.23 présente le résultat du modèle de CART. Ainsi, la répartition des groupes est obtenue via CART.

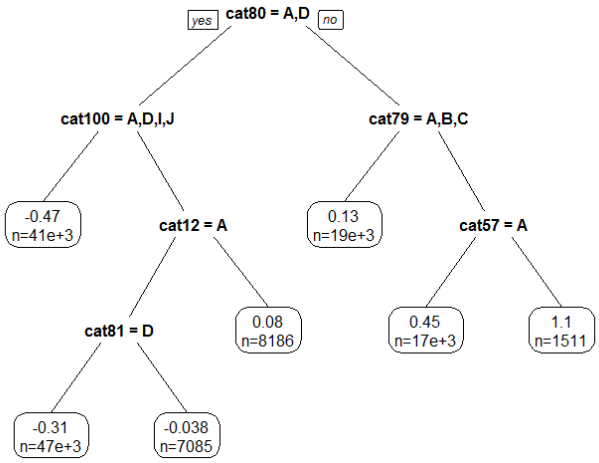

FIGURE 4.23 – Résultat du modèle de CART

5. **Prédiction de l'appartenance aux groupes.** Une fois le modèle CART calibré, il est utilisé pour prédire l'appartenance aux groupes sur l'échantillon de test. Plus précisément, chaque fois que nous fournissons à CART les caractéristiques x_i d'un individu i comme l'entrée, nous trouvons la feuille où se situe x_i à la sortie de CART.

6. **Calibrage d'une régression groupe par groupe via XGBoost.** Revenons sur l'échantillon d'apprentissage à présent. Les modèles de régression d'XGBoost sont calibrés respectivement sur chaque groupe. L'XGBoost construit sur le groupe k est noté XGB_k.
7. **Prédiction sur l'échantillon de test.** Finalement, sur l'échantillon de test, avec une observation x_j donnée, les modèles d'XGBoost calibrés dans l'étape précédente sont servis pour faire des prédictions :
   si $x_j \in$ Groupe k selon la prédiction de CART à l'étape 5, alors sa prédiction de montants des sinistres vaudra :

$$\hat{y}_j = \mathrm{XGB}_k(x_j).$$

L'arbre complet issu de CART possède 7 feuilles. Autrement dit, nous obtenons 7 groupes. Cet arbre complet est ensuite élagué de sorte que le nombre de feuilles varie de 2 à 6. Ainsi nous pouvons comparer les performances des modèles selon les nombres différents de groupes.

Nous avons inclus cette méthode dans l'approche *a priori* au lieu de l'approche *a posteriori* car il n'y a pas eu lieu un calibrage du modèle à proprement parler puisque les prédictions sont toutes égales à la moyenne des montants des sinistres dans l'échantillon d'apprentissage.

4.2.2.6 6ème méthode : CART pour prédire les montants des sinistres

Algorithme

1. **Séparation des échantillons d'apprentissage et test.** L'ensemble de l'échantillon est séparé en échantillon d'apprentissage et échantillon de test.
2. **Calibrage du modèle de régression via CART.** Sur l'échantillon d'apprentissage, nous calibrons un modèle de CART, en prenant les caractéristiques des assurés X comme entrée et le montant des sinistres comme variable à prédire. Comme nous l'avons mentionné à la sous-section 3.2.1 CART, à l'issue de CART, les observations au sein d'une même feuille peuvent être vues comme des risques relativement homogènes. Ainsi, les groupes sont obtenus via CART.
3. **Prédiction de l'appartenance aux groupes via CART.** Une fois le modèle de CART calibré, il peut être utilisé pour prédire l'appartenance aux groupes sur l'échantillon de test. Plus précisément, chaque fois que nous

fournissons à CART les caractéristiques x_j d'un individu j comme entrée, nous pouvons trouver la feuille où se situe x_j à la sortie de CART.

4. **Calibrage d'une régression groupe par groupe via XGBoost.** Revenons sur l'échantillon d'apprentissage à présent. Les modèles de régression d'XGBoost sont calibrés respectivement sur chaque groupe. L'XGBoost construit sur le groupe k est noté XGB_k.
5. **Prédiction sur l'échantillon de test.** Finalement, sur l'échantillon de test, avec une observation x_j donnée, Nous pouvons se servir des modèles d'XGBoost calibrés dans l'étape précédente et l'appartenance aux groupes de l'étape 3 pour faire des prédictions : si $x_j \in$ Groupe k selon la prédiction de CART à l'étape 3, alors sa prédiction de coût de sinistres vaudra :

$$\hat{y}_j = \mathrm{XGB}_k(x_j).$$

L'arbre complet issu de CART possède 9 feuilles. Autrement dit, nous obtenons 9 groupes. Cet arbre complet est ensuite élagué de sorte quele nombre de feuilles varie de 2 à 8. Ainsi nous pouvons comparer les performances des modèles selon les nombres différents de groupes.

4.2.2.7 7ème méthode : GMM sur log(montants des sinistres)

Comme ce qui est mentionné à sous-section 2.1.4 Hétérogénéité des risques et modèle de mélange de lois et sous-sous-section 3.3.2.3 Modèle de mélange gaussien, l'hétérogénéité des risques peut se modéliser statistiquement par les modèles de mélange de lois, il s'avère alors naturel de penser à effectuer une classification non-supervisée via un modèle de mélange de lois. Toutefois, l'estimation des paramètres de chaque loi composante reste toujours un sujet très difficile à traiter. Et donc en pratique seuls les mélanges de lois gaussiennes sont considérés. Cela constitue l'une des raisons principales que nous optons d'utiliser le GMM. Comme la variable montant des sinistres ne prend que des valeurs positives, nous décidons donc de la passer en logarithme pour que sa densité ressemble plus à celle d'une loi normale. La transformation en logarithme est souvent pratiquée en concours de *data-science* comme *Kaggle*. D'autres transformations sont aussi envisageables. Les lecteurs intéressés sont invités à expérimenter.

Algorithme

1. **Séparation des échantillons d'apprentissage et test.** L'ensemble d'échantillon est séparé en l'échantillon d'apprentissage et l'échantillon de test.
2. **Classification des log(montants des sinistres) via GMM.** Sur l'échantillon d'apprentissage, une classification non-supervisée sur log(montants des sinistres) est effectuée via le modèle de mélanges gaussiens (GMM).

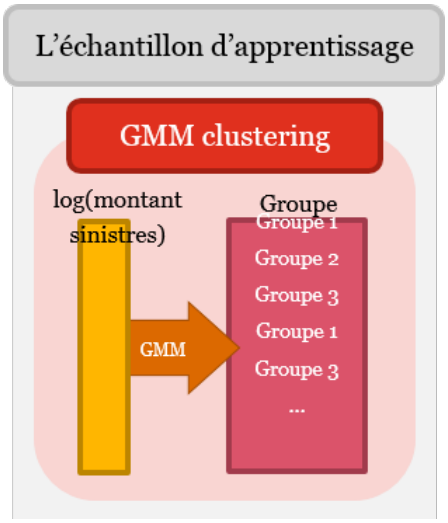

FIGURE 4.24 – L'étape 2 : Classification des log(montants des sinistres) via GMM

3. **Calibrage du modèle de classification via XGBoost.** Étant donné l'appartenance aux groupes et la matrice des variables explicatives X, un modèle de XGBoost classification est ensuite calibré sur l'échantillon d'apprentissage afin que ce modèle puisse prédire les probabilités qu'une observation appartienne à un groupe.
4. **Prédiction par le modèle de classification via XGBoost.** Une fois le modèle de classification calibré, il peut se servit pour prédire les probabilités d'appartenance aux groupes sur le test set. Plus précisément, chaque fois que nous fournissons les caractéristiques x_i d'un individu i comme entrée aux modèle, le modèle donnera un vecteur de probabilités dont la 1ère coordonnée est la probabilité que x_i appartiennent au groupe 1, la 2ème coordonnée est la probabilité que x_i soit inclus dans le groupe 2, ainsi de suite.
5. **Calibrage d'une régression groupe par groupe via XGBoost.** Revenons sur l'échantillon d'apprentissage à présent. Un modèle de régression d'XGBoost est calibré respectivement sur chaque groupe. Par exemple, pour

le groupe k, l'XGBoost va prendre X des observations au sein du groupe k comme entrée et essayer de prédire les montants des sinistres de ces observations. L'XGBoost construit sur le groupe k se note XGB_k.

6. **Prédiction sur l'échantillon de test.** Finalement, sur l'échantillon de test, avec une observation x_j donné, les modèles d'XGBoost calibrés dans l'étape précédente sont utilisés pour faire des prédictions :
   - Si nous supposons que $x_j \in$ groupe 1, la prédiction sera $\text{XGB}_1(x_j)$;
   - Si nous supposons que $x_j \in$ groupe 2, la prédiction sera $\text{XGB}_2(x_j)$;
   - ...
   - Si nous supposons que $x_j \in$ groupe k, la prédiction sera $\text{XGB}_k(x_j)$;
   - ...

   De plus, nous supposons qu'au total il y a K groupes et nous désignons les probabilités d'appartenance aux groupes obtenues via l'XGBoost classification à l'étape 6 par $\hat{\mathbf{P}}(x_j \in \text{groupe } 1), \ldots, \hat{\mathbf{P}}(x_j \in \text{groupe } k), \ldots, \hat{\mathbf{P}}(x_j \in \text{groupe } K)$, alors la prédiction de montant des sinistres pour l'observation x_j vaudra :

$$\hat{y}_j = \sum_{k=1}^{K} \hat{\mathbf{P}}(x_j \in \text{groupe } k) \cdot \text{XGB}_k(x_j).$$

Cette méthode ressemble beaucoup à la première méthode de l'approche *a posteriori* décrite à la sous-sous-section 4.2.1.1. Précisons quelques grandes différences entre les deux méthodes :

- Premièrement, cette méthode est dans l'approche *a priori* alors que l'autre est dans l'approche *a posteriori* ;
- Deuxièmement, bien que le GMM soit mis en œuvre dans les deux méthodes, mais il est effectué sur log(montant des sinistres) dans cette méthode alors qu'il est appliqué sur les résidus de déviance dans l'autre méthode.

4.2.2.8 8ème méthode : GMM sur les résidus obtenus par le modèle naïf

Cette méthode est une variante de la première méthode de l'approche *a posteriori* décrite à la sous-sous-section 4.2.1.1. Dans celle-ci, au lieu de calibrer un modèle d'XGBoost sur l'ensemble des données à l'étape 2 nous définissons un

modèle naïf. Le terme “naïf” est employé car toutes les prédictions de ce modèle seront pareilles : elles seront égales à la moyenne des montants des sinistres dans l'échantillon. Le reste de l'algorithme reste identique. Nous ne le détaillons donc pas ici. Nous avons inclus cette méthode dans l'approche *a priori* au lieu de l'approche *a posteriori* car il n'a pas eu lieu un vrai calibrage d'un modèle.

4.3 Comparaison des résultats

Dans cette section, nous présenterons les résultats obtenus avec différentes méthodes sur le même échantillon de test. Pour chaque méthode (sauf CART), nous avons fait varier le nombre de groupes de 2 à 10 pour que nous puissions comparer les performances selon le nombre de groupes et trouver le nombre optimal de groupes. Pour les méthodes impliquant CART, nous élaguons l'arbre pour faire varier le nombres de groupes. Le nombre de groupes n'excède pas 10 pour deux raisons : le temps et la mémoire de l'ordinateur sont limités. De plus, il n'est pas non plus réaliste de calibrer plus de 10 modèles différents sur un même portefeuille en pratique.

Rappelons que notre objectif consiste à détecter des groupes dans le but de mieux prédire la sinistralité grâce à cette segmentation, ainsi que les trois mesures d'évaluation évoquées à la section 2.2 Objectif :

- La déviance Gamma ;
- L'erreur quadratique moyenne, ou *mean squared error* (MSE) en anglais ;
- L'erreur absolue moyenne, ou *mean absolute error* (MAE) en anglais.

Ces trois mesures seront utilisées non seulement pour déterminer le nombre optimal de groupes, mais aussi pour déterminer la méthode la plus adéquate.

4.3.1 Résultats de l'approche *a posteriori*

Le tableau ci-dessous (Figure 4.25) montre les résultats des différentes méthodes dans l'approche *a posteriori*. La première ligne (nombre de groupes égal à 1) correspond aux résultats lorsque nous calibrons un modèle de régression d'XGBoost sur l'ensemble des données sans aucune segmentation.

	GMM sur résidus			K-means sur résidus		
# groupes	Déviance	MSE	MAE	Déviance	MSE	MAE
1	14391	6662785	1289	14391	6662785	1289
2	13898	4771667	1239	13684	5382980	1219
3	13655	4119450	1212	**13644**	**4096884**	**1211**
4	**13573**	4405684	**1209**	13698	4698546	1219
5	14145	5833586	1277	13812	4626776	1230
6	13632	4305397	1215	13917	4887962	1242
7	13657	**4102228**	1214	13886	4646197	1238
8	13870	4481111	1240	13691	4166314	1218
9	14027	4813846	1264	13995	5031864	1256
10	14126	5192407	1277	13780	4726072	1230

FIGURE 4.25 – Résultats des méthode de l'approche *a posteriori*

	GMM sur résidus			K-means sur résidus		
# groupes	Déviance	MSE	MAE	Déviance	MSE	MAE
1	0%	0%	0%	0%	0%	0%
2	3%	28%	4%	5%	19%	5%
3	5%	38%	6%	**5%**	**39%**	**6%**
4	**6%**	34%	**6%**	5%	29%	5%
5	2%	12%	1%	4%	31%	5%
6	5%	35%	6%	3%	27%	4%
7	5%	**38%**	6%	4%	30%	4%
8	4%	33%	4%	5%	37%	5%
9	3%	28%	2%	3%	24%	3%
10	2%	22%	1%	4%	29%	5%

FIGURE 4.26 – Résultats des méthode de l'approche *a posteriori* (présentés en diminution de pourcentage)

Nous remarquons une diminution de la déviance (Figure 4.27), de MSE (Figure 4.28) et de MAE (Figure 4.29) significative lorsque nous segmentons le portefeuille en plusieurs groupes. La déviance atteint son minimum lorsque le nombre de groupes est égal à 4 avec la méthode de "GMM sur les résidus de déviance". Il en est de même pour le MAE. En conséquence, pour l'approche *a posteriori*, nous retenons la méthode de "GMM sur les résidus de déviance", avec un nombre de groupes égal à 4. Cela nous permet d'obtenir une diminution de 6% de la déviance, de 34% de MSE et de 6% de MAE.

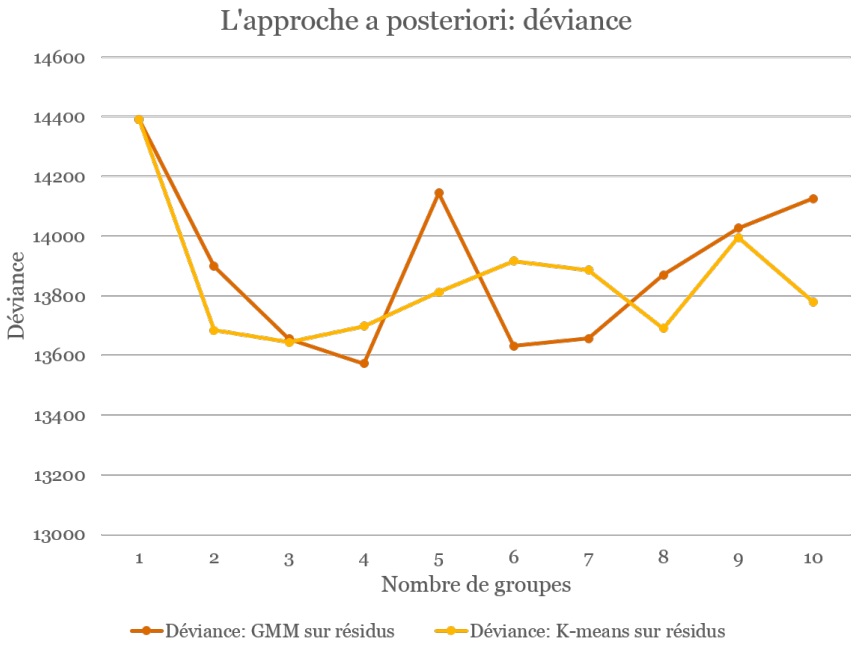

FIGURE 4.27 – Déviances des méthodes de l'approche *a posteriori*

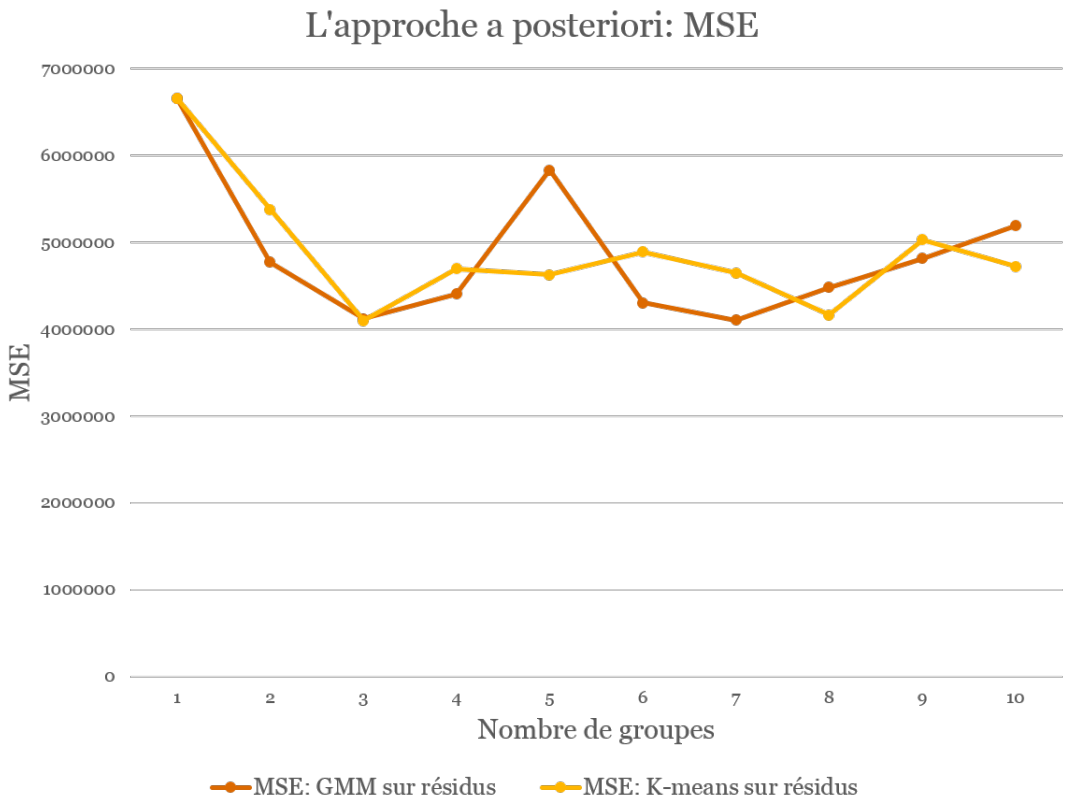

FIGURE 4.28 – MSE des méthodes de l'approche *a posteriori*

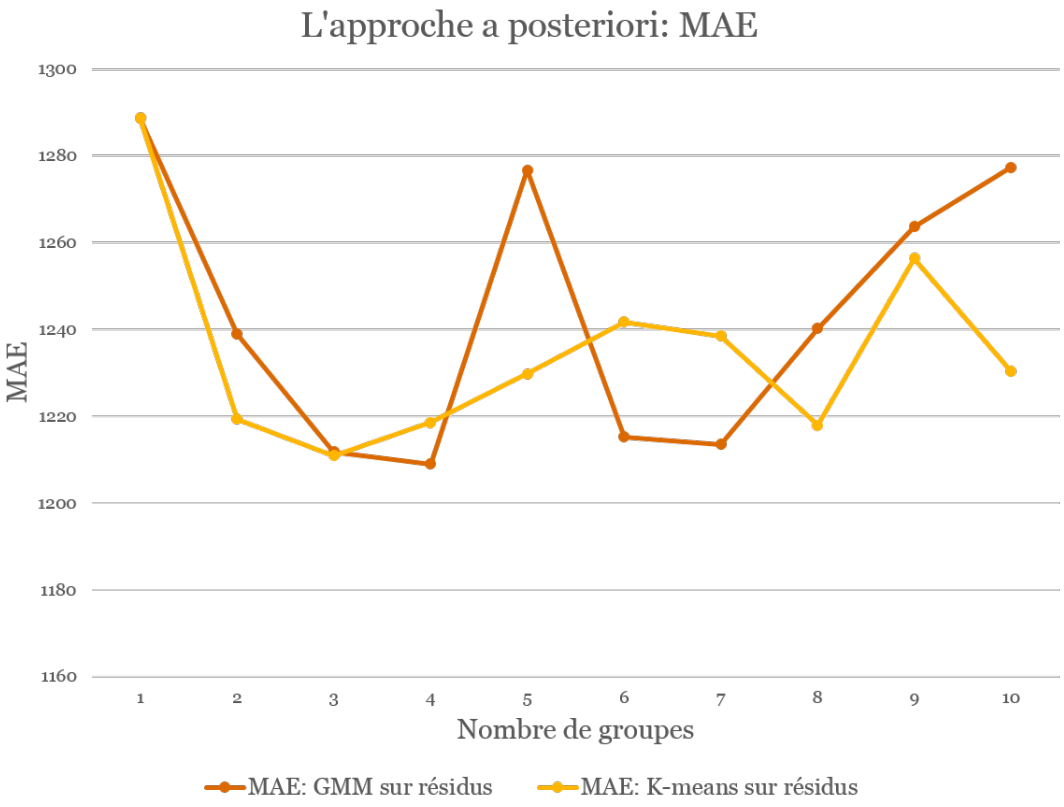

FIGURE 4.29 – MAE des méthodes de l'approche *a posteriori*

4.3.2 Résultats de l'approche *a priori*

	ACP - K-means			Auto-encodeur - K-means			ACP - SOM			ACP - SOM - K-means			CART sur résidus du modèle naïf			CART pour prédire les montants des sinistres			GMM sur log(montant des sinistres)			GMM sur résidus du modèle naïf		
# groupes	Déviance	MSE	MAE	Déviance	MSE	MAE	Déviance	MSE	MAE	Déviance	MSE	MAE	Déviance	MSE	MAE	Déviance	MSE	MAE	Déviance	MSE	MAE	Déviance	MSE	MAE
1	14391	6662785	1289	14391	6662785	1289	14391	6662785	1289	14391	6662785	1289	14391	6662784	1289	14391	6662785	1289	14391	6662785	1289	14391	6662785	1289
2	14377	5334584	1281	14403	5885590	1286	14412	5717150	1286	14401	4818474	1281	13945	7602815	1232	13945	7602815	1232	13672	4017284	1203	13653	4115283	1200
3	14092	4641201	1236	14198	6697378	1265	14419	5852847	1286	14411	4832592	1284	13959	7598894	1230	13927	4793227	1222	13450	3761333	1191	13454	4235899	1191
4	**14049**	5461240	**1235**	14192	5000251	1260	14412	4784946	1283	14407	4879511	1277	13936	4272848	1221	13923	4011458	1217	**13439**	3790181	**1188**	**13447**	3819091	**1185**
5	14140	4704327	1244	**14177**	7859406	1265	14420	5892828	1284	14410	4931970	1280	13867	4230564	1216	13923	3848361	1217	13571	**3702861**	1192	13628	3800215	1196
6	14172	**4383126**	1242	14213	4933398	1255	14332	4412087	1263	14374	4845397	1275	**13860**	**4100761**	1213	**13838**	**3818105**	**1210**	13580	3722412	1194	13507	**3699838**	1189
7	14090	4639785	1239	14249	5203130	1254	14475	6048352	1287	14377	**4413957**	**1267**	13890	4106431	**1212**	13850	3840212	1212	13668	3743355	1197	13795	3795146	1204
8	14122	5329864	1239	14262	6779059	1261	14372	4535029	1269	14482	4697304	1279				13866	3867444	1213	13528	3736802	1191	13750	3750723	1202
9	14126	4739845	1238	14338	**4692820**	1256	14392	**4288411**	**1260**	**14371**	4618432	1273				13878	3866375	1213	13783	3792718	1203	14040	3821620	1218
10	14126	4485909	1238	14236	5649391	**1246**	**14280**	4420387	1263	14378	4515491	1270							13645	3815938	1195	13621	3813418	1195

FIGURE 4.30 – Résultats des méthode de l'approche *a priori*

	ACP - K-means			Auto-encodeur - K-means			ACP - SOM			ACP - SOM - K-means			CART sur résidus du modèle naïf			CART pour prédire les montants des sinistres			GMM sur log(montant des sinistres)			GMM sur résidus du modèle naïf		
# groupes	Déviance	MSE	MAE	Déviance	MSE	MAE	Déviance	MSE	MAE	Déviance	MSE	MAE	Déviance	MSE	MAE	Déviance	MSE	MAE	Déviance	MSE	MAE	Déviance	MSE	MAE
1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
2	0%	20%	1%	0%	12%	0%	0%	14%	0%	0%	28%	1%	3%	-14%	4%	3%	-14%	4%	5%	40%	7%	5%	38%	7%
3	2%	30%	4%	1%	-1%	2%	0%	12%	0%	0%	27%	0%	3%	-14%	5%	3%	28%	5%	7%	44%	8%	7%	36%	8%
4	**2%**	18%	**4%**	1%	25%	2%	0%	28%	0%	0%	27%	1%	3%	36%	5%	3%	40%	6%	**7%**	43%	**8%**	**7%**	43%	**8%**
5	2%	29%	3%	**1%**	-18%	2%	0%	12%	0%	0%	26%	1%	4%	37%	6%	3%	42%	6%	6%	**44%**	8%	5%	43%	7%
6	2%	**34%**	4%	1%	26%	3%	0%	34%	2%	0%	27%	1%	**4%**	**38%**	6%	**4%**	**43%**	**6%**	6%	44%	7%	6%	**44%**	8%
7	2%	30%	4%	1%	22%	3%	-1%	9%	0%	0%	**34%**	**2%**	3%	38%	**6%**	4%	42%	6%	5%	44%	7%	4%	43%	7%
8	2%	20%	4%	1%	-2%	2%	0%	32%	2%	-1%	29%	1%				4%	42%	6%	6%	44%	8%	4%	44%	7%
9	2%	29%	4%	0%	**30%**	2%	0%	**36%**	**2%**	**0%**	31%	1%				4%	42%	6%	4%	43%	7%	2%	43%	5%
10	2%	33%	4%	1%	15%	**3%**	**1%**	34%	2%	0%	32%	1%							5%	43%	7%	5%	43%	7%

FIGURE 4.31 – Résultats des méthode de l'approche *a priori* (présentés en diminution de pourcentage)

Le tableau ci-dessus (Figure 4.30) montre les résultats de différentes méthodes dans l'approche *a priori*. La première ligne (nombre de groupes égal à 1) correspond toujours aux résultats lorsque nous calibrons un modèle de régression d'XGBoost sur l'ensemble des données sans aucune segmentation.

Encore une fois nous remarquons une diminution significative de la déviance (Figure 4.32), de MSE (Figure 4.33) et de MAE (Figure 4.34) lorsque nous segmentons le portefeuille en plusieurs groupes. Parmi les 8 méthodes de l'approche *a priori*, la méthode de "GMM sur log(montant des sinistres)" s'avère la plus performante en terme de la déviance, MSE et MAE. La déviance est optimisée lorsque le nombre de groupes est égal à 4. Il en est de même pour le MAE. Ce nombre optimal est également cohérent avec celui de l'approche *a posteriori*. Cela nous invite à retenir la méthode de "GMM sur log(montant des sinistres)" avec le nombre de groupes égal à 4 pour l'approche *a priori*. Cela nous permet d'obtenir une diminution de 7% de la déviance, de 43% de MSE et de 8% de MAE.

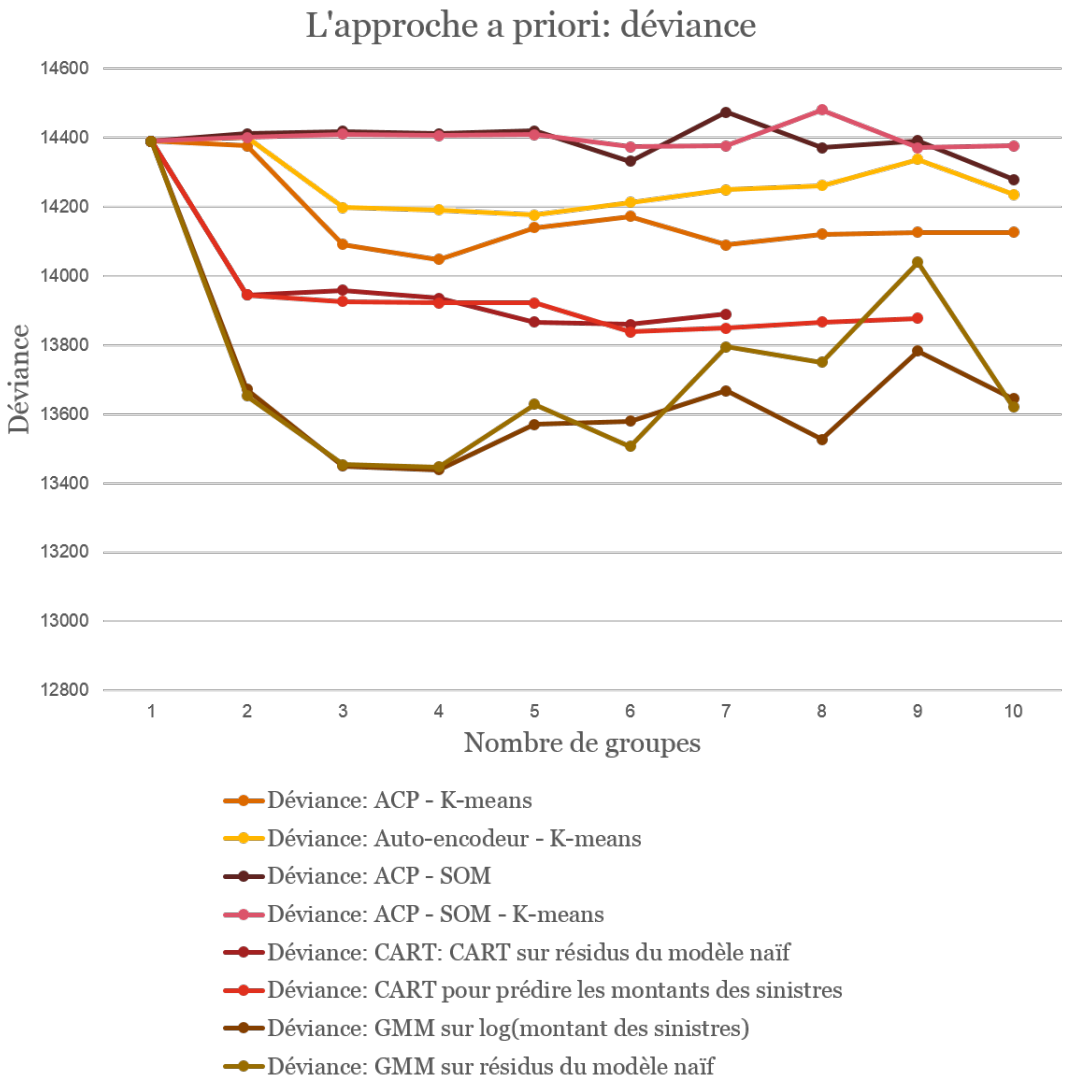

FIGURE 4.32 – Déviances des méthodes de l'approche *a priori*

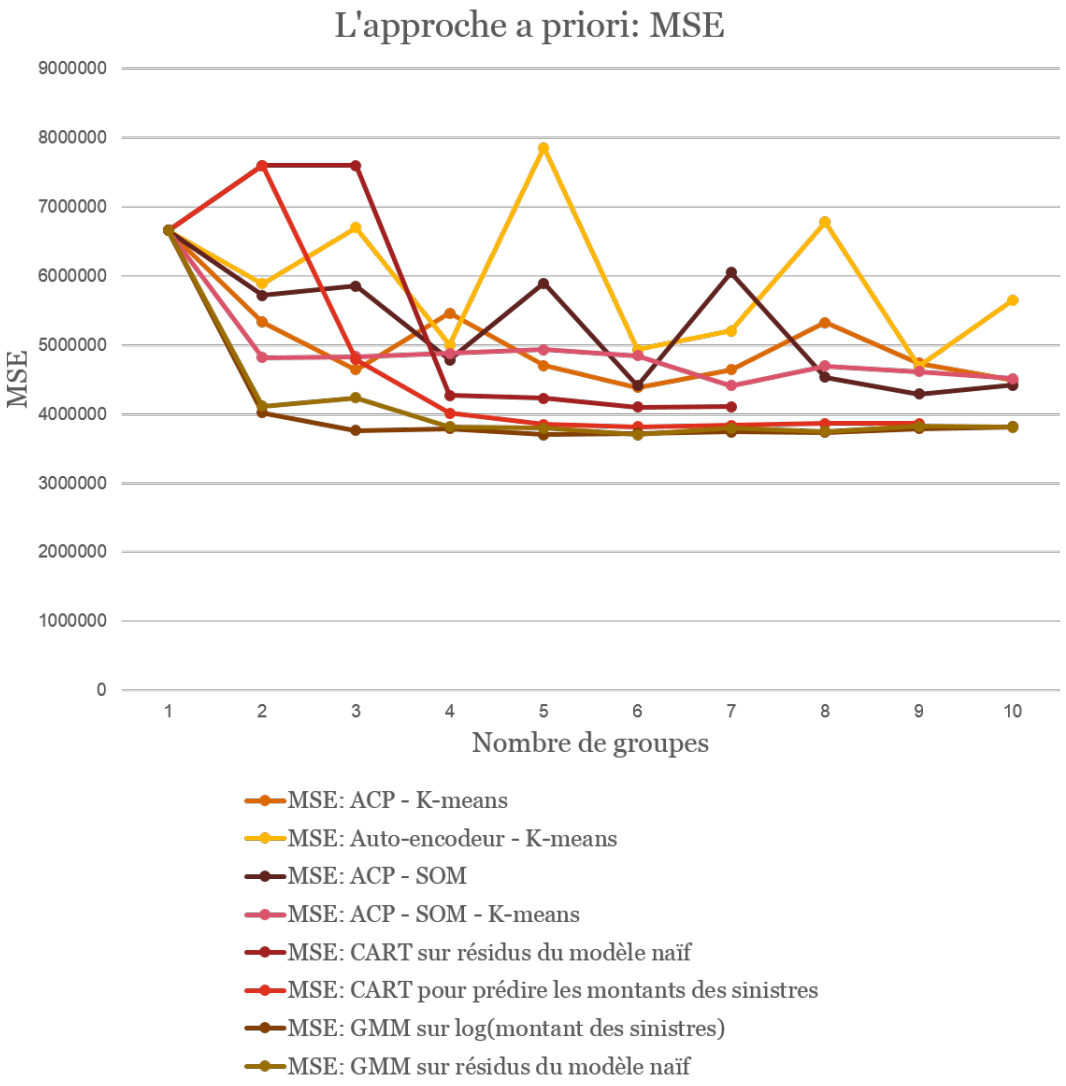

FIGURE 4.33 – MSE des méthodes de l'approche *a priori*

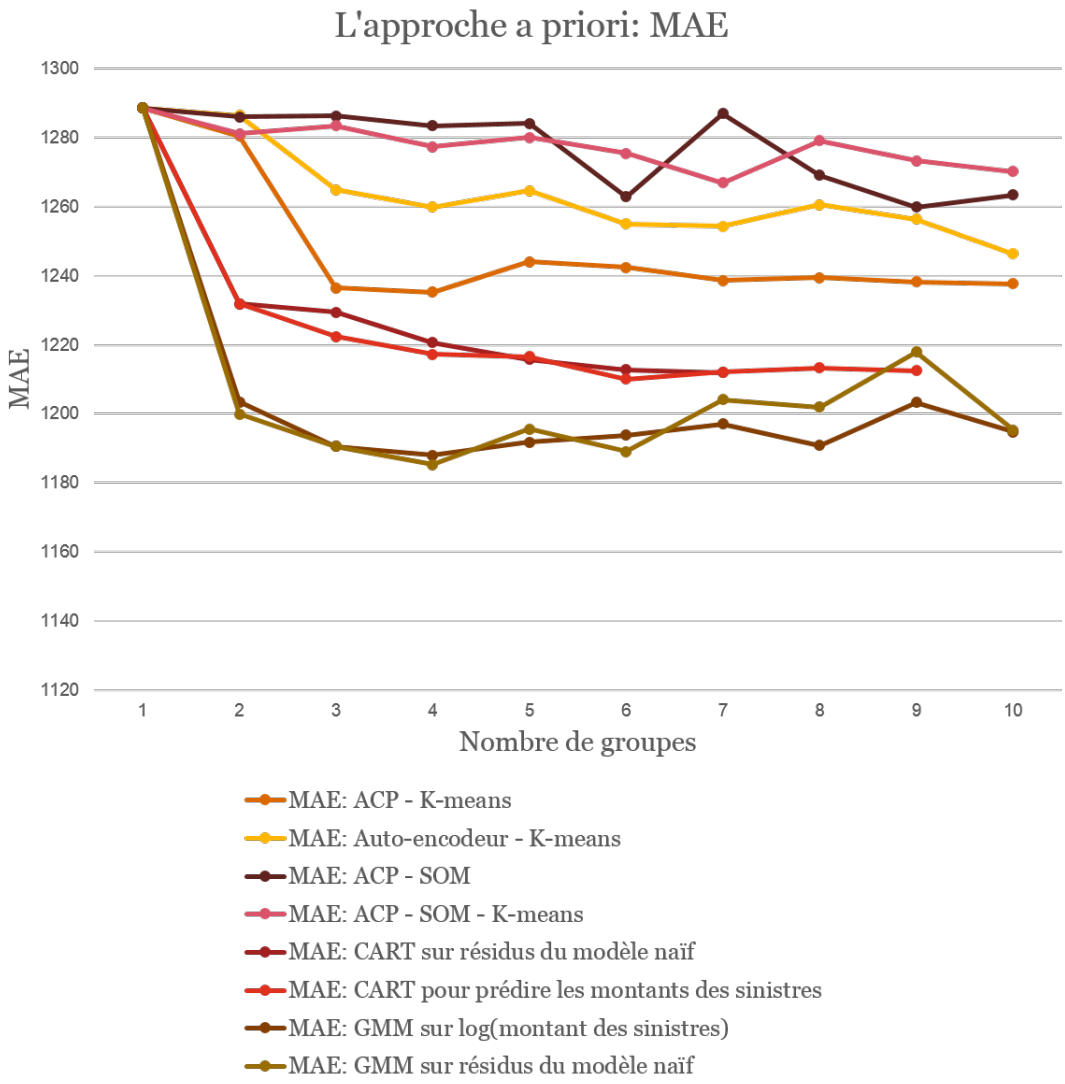

FIGURE 4.34 – MAE des méthodes de l'approche *a priori*

4.3.3 Comparaison des résultats de l'approche *a posteriori* et l'approche *a priori*

La Figure 4.35, la Figure 4.36 et la Figure 4.37 comparent les résultats issus des différentes méthodes des deux approches.

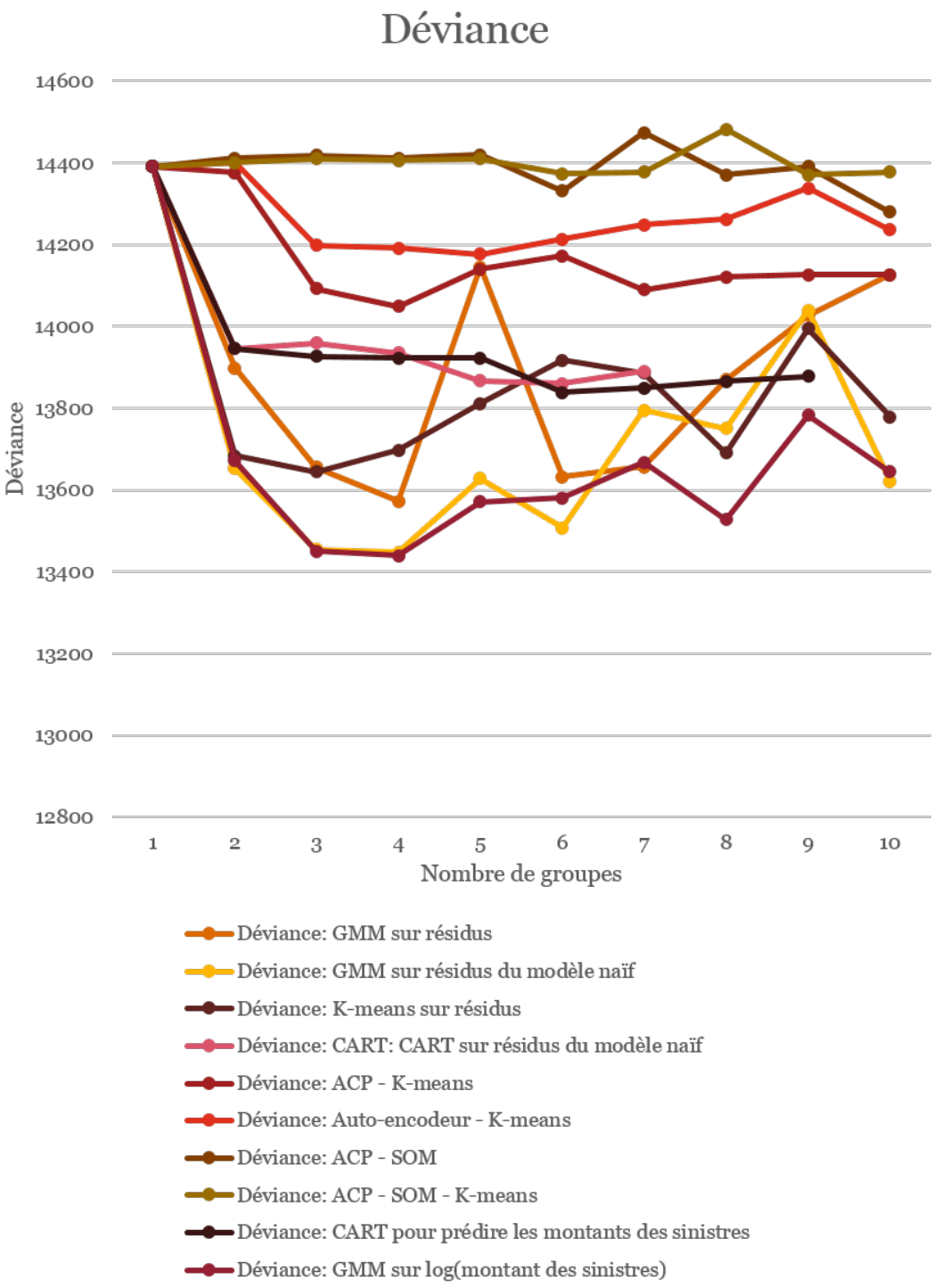

FIGURE 4.35 – Déviances des deux approches

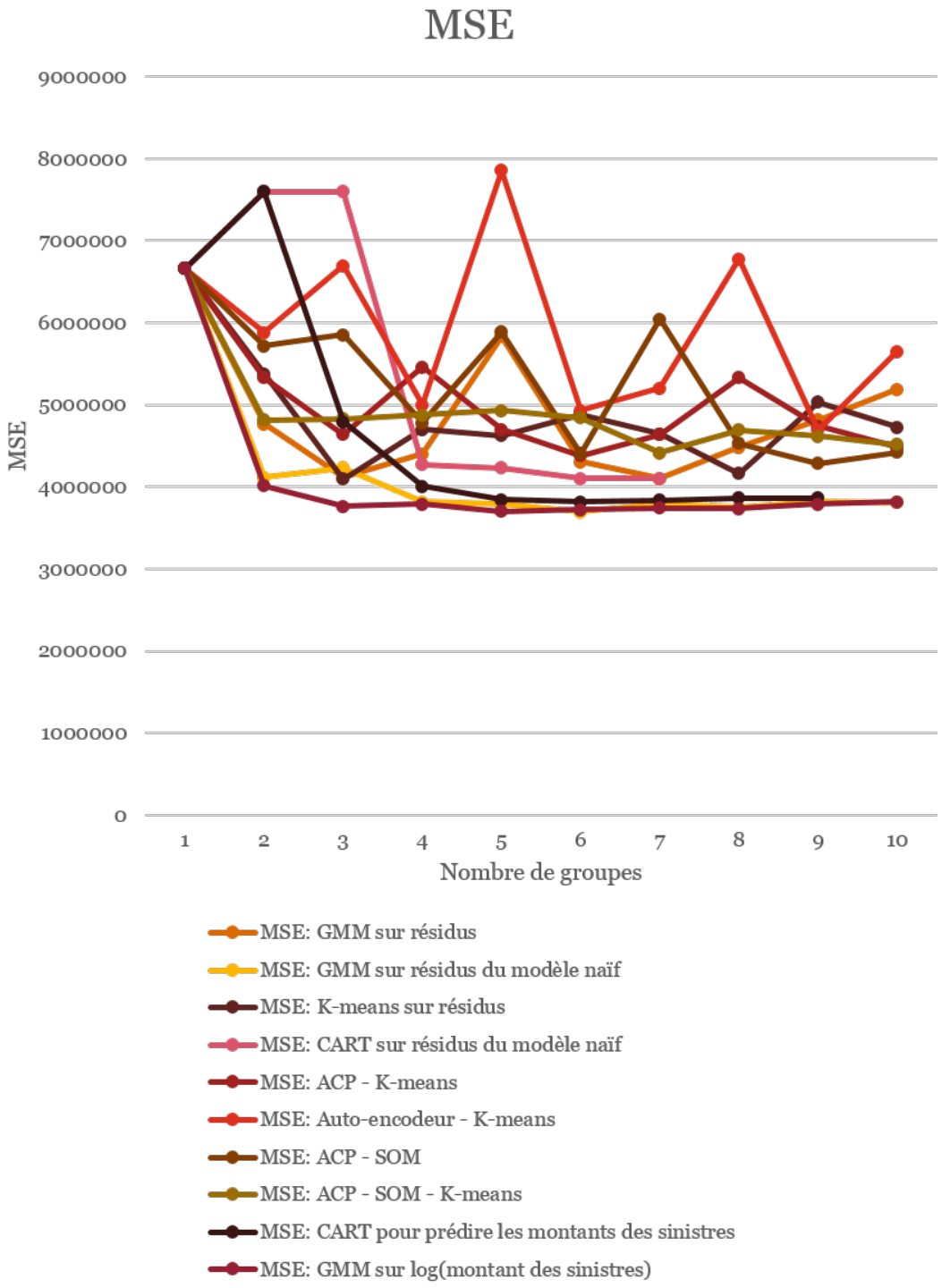

FIGURE 4.36 – MSE des deux approches

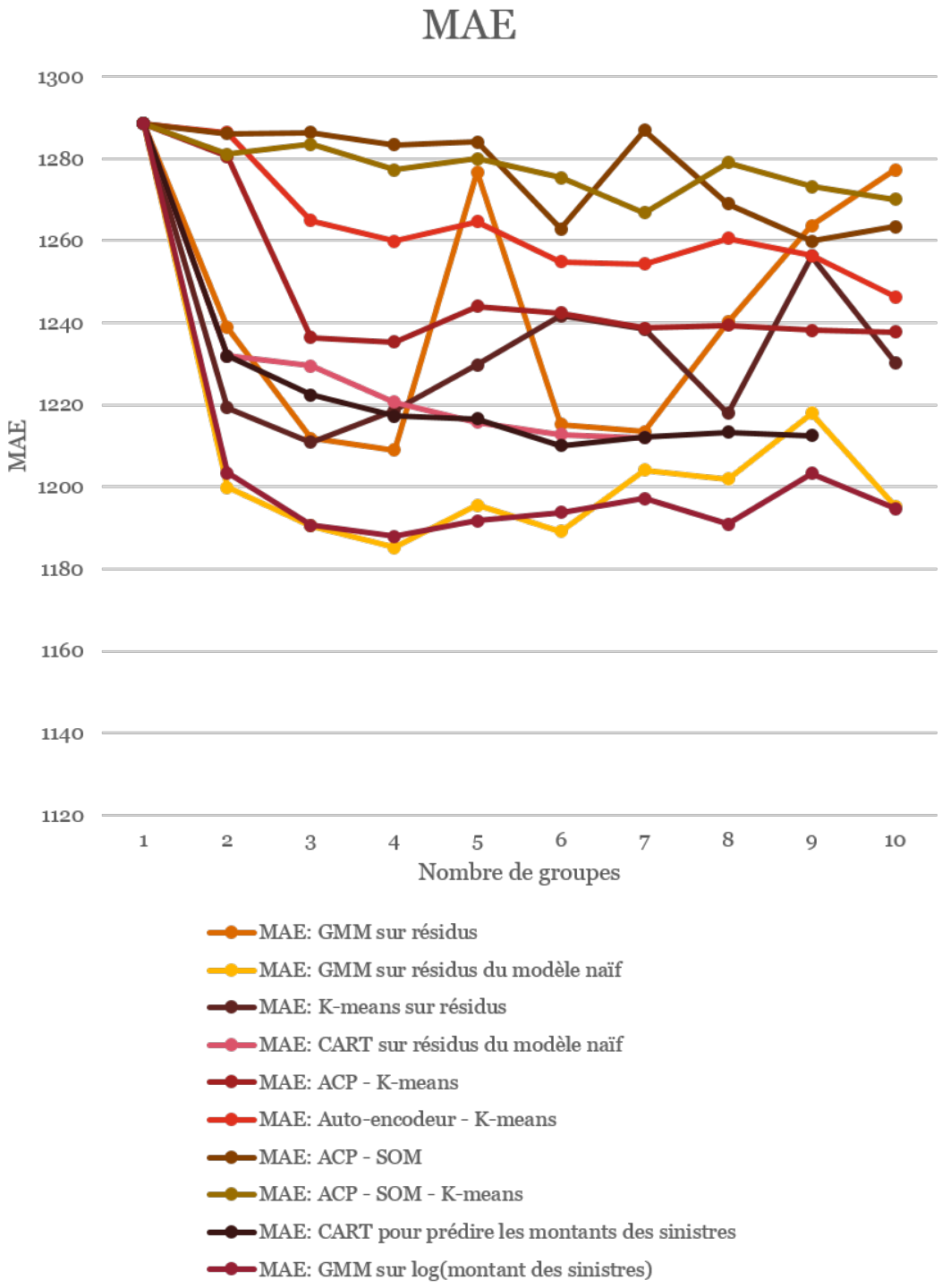

FIGURE 4.37 – MAE des deux approches

Nous constatons que globalement la méthode de "GMM sur log(montant des sinistres)" de l'approche *a priori* donne la meilleure prédiction en terme de la déviance, MSE et MAE, ce qui nous invite à retenir cette méthode pour identifier les groupes de risques homogènes. Le nombre optimal de groupes est égal à 4.

Ces résultats nous montre que non seulement une segmentation en groupes homogènes permet de prédire au plus près les sinistres, de plus, la segmentation obtenue via l'approche *a priori* peut faire encore plus judicieusement que celle-ci obtenue via l'approche *a posteriori*, ce qui nous évite toutes les démarches laborieuses et coûteuses de l'approche *a posteriori*.

Chapitre 5

Conclusion et perspectives

Il est temps de faire le bilan sur ce que nous avons parcouru dans ce mémoire. Nous avons commencé par exposer les problématiques : l'hétérogénéité des risques peut potentiellement remettre en question les hypothèses de la distribution des risques et ainsi la robustesse du modèle sous-jacent. Face à un portefeuille hétérogène, il est donc nécessaire de le partitionner en groupes de risques homogènes, de sorte qu'au sein de chaque groupe les risques soient identiquement distribués. Pour ce sujet, deux approches sont proposées : l'approche *a posteriori* et l'approche *a priori*. Nous avons ensuite revisité les théories de certains algorithmes supervisés et non-supervisés. Différentes méthodes sont développées en s'appuyant sur ces algorithmes d'apprentissage automatique. Elles sont alors mises en application pour aboutir à la finalité recherchée : détecter les groupes de risques homogènes d'une manière automatique dans un but de mieux prédire la sinistralité grâce à cette segmentation.

Les résultats obtenus sont prometteurs et correspondent à nos attentes. Une segmentation en groupes de risques homogènes s'est avérée être propice à mieux prédire la sinistralité. Nous avons obtenu une diminution significative de la déviance, MSE et MAE grâce à la segmentation, que ce soit via l'approche *a posteriori*, ou via l'approche *a priori*. Parmi toutes les méthodes proposées dans les deux approches, globalement la méthode "GMM sur log(montant des sinistres)" dans l'approche *a priori* réalise les meilleures performances en termes de déviance, MSE et MAE, ce qui nous invite à porter notre choix pour cette méthode. Cela signifie qu'*a priori*, avant le calibrage du modèle, nous pouvons déjà déterminer une segmentation adaptée avec l'aide de cette méthode.

Limites de l'étude et axes d'amélioration

Faute de temps et de performance de calcul, nous n'avons pas pu explorer toutes les pistes envisageables. Citons par exemple l'algorithme t-SNE (*t-distributed stochastic neighbor embedding*) pour réduire la dimension, DBSCAN (*density-based spatial clustering of applications with noise*) et la classification ascendante hiérarchique pour faire classification non-supervisée. Nous pouvons les expérimenter avec un serveur plus puissant dans le futur.

Parmi les algorithmes que nous avons pu tester, d'autres combinaisons des étapes sont aussi concevables. Par exemple, dans l'approche *a priori*, nous avons mis en œuvre la méthode "ACP – SOM" . Mais nous pouvons aussi mettre en place "auto-encodeur – SOM".

Par ailleurs, nous pouvons également expérimenter d'autres paramètres pour certains algorithmes. Dans l'approche *a priori*, nous avons implémenté une version simple de l'auto-encodeur, avec une couche cachée contenant deux neurones. Nous pouvons mettre en œuvre d'autres versions plus complexes de l'auto-encodeur, par exemple, avec trois couches cachées contenant plus de neurones. Il en est de même pour l'algorithme SOM. Nous avons choisi une grille de taille 50×50. Nous pouvons tester d'autres grilles de taille différente.

Nous pouvons aussi améliorer notre étude en enrichissant la base de données. Comme le dit le démographe américain James W. Vaupel : "*Toutes les populations sont hétérogènes : deux individus de même âge et de même sexe dans une population peuvent présenter deux risques de décès très différents.*" Il existe de nombreux facteurs de risque qui ne sont pas observables ou mesurables. Citons par exemple, la consommation d'alcool, l'utilisation de portable au volant et le manque de sommeil. Si les variables explicatives comme âge, éducation, profession ne nous disent pas tout sur le profil de risque, l'historique de sinistralité peut nous révéler plus d'information. En prenant en compte l'historique de sinistralité, nous pouvons éventuellement identifier les groupes encore plus homogènes.

En outre, dans la base de données étudiée, en outre des variables explicatives comme les caractéristiques des assurés, nous ne disposons que d'une variable à prédire - le montants des sinistres. Or cette variable elle-même est très volatile car sa valeur dépend non seulement des caractéristiques des assurés, mais aussi des facteurs purement dûs au hasard. Contrairement à la variable "montants des sinistres", la variable "nombre de sinistres" est peu influencée par les facteurs purement dus au hasard. En l'ajoutant dans la base de données à étudier, nous pouvons développer une méthodologie plus pertinente.

D'autre part, pour des raisons de confidentialité, la base de données à notre disposition est anonymisée. Il est donc difficile de caractériser clairement les groupes de risques homogènes trouvés.

D'ailleurs, afin d'estimer la fiabilité de nos méthodes, il est souhaitable d'implémenter la "k-fold validation croisée" (*k-fold cross validation*). Il y a plusieurs façons de validation croisée. Dans ce mémoire, nous avons mis en place une version simple de validation croisée. Nous avons séparé notre base de données en deux sous-échantillons : l'échantillon d'apprentissage et l'échantillon de test. Nous calibrons le modèle sur l'échantillon d'apprentissage et la performance du modèle est mesurée sur l'échantillon de test. Dans l'idéal, comme notre base de données est suffisamment grande, avec 180 000 lignes, nous pouvons implémenter un 5-fold validation croisée pour obtenir des résultats plus fiables. D'autre part, nous pouvons également expérimenter nos méthodes sur plusieurs bases de données différentes et en retirer une conclusion plus convaincante.

La transformation des variables qualitatives en variables quantitatives est aussi essentielle. Dans cette étude, nous n'avons expérimenté qu'une seule méthode - "l'espérance de la variable à prédire". Toutefois, nous pouvons rechercher d'autres choix plus judicieux pour l'avenir.

Enfin, dernier élément mais non des moindres, la détection du nombre optimal des groupes homogènes reste toujours une problématique à résoudre. Dans cette étude, il est déterminé *a posteriori* des calibrages des modèles de régression. Les calibrages des modèles étant coûteux, il est alors désirable de le déterminer et d'identifier les groupes de risques homogènes *a priori* de la modélisation.

Si ce mémoire est désormais terminé, les méthodes développées ici peuvent avoir un vaste champ d'application et ne concernent pas seulement la tarification en non-vie, mais aussi dans tout type de modélisation de risques, que ce soit dans le provisionnement ou dans le calcul du SCR en Solvabilité II. Le chemin qui reste à parcourir est encore long !

Bibliographie

[1] Michel Denuit, Arthur Charpentier, *Mathématiques de l'assurance non-vie, Tome I : Principes Fondamentaux de Théorie du Risque*, Economica, 2004.

[2] Michel Denuit, Arthur Charpentier, *Mathématiques de l'assurance non-vie, Tome II : Tarification et provisionnement*, Economica, 2009.

[3] Dan Tevet, *And the winner is... ? How to pick a better model : Part 2 Goodness-of-fit and internal stability*, Verisk Insurance Solutions.

[4] Gilles Dupin, Alain Monfort, Jean-Pierre Verle, *Robust inference in rating models*, Proceedings of the 34th ASTIN Colloquium, 2003.

[5] Antoine Paglia, Martial V.PHELIPPE-GUINVARC'H, *Tarification des risques en assurance non-vie, une approche par modèle d'apprentissage statistique*, Bulletin Français d'Actuariat, vol. 11, numéro 22, juillet - décembre 2011, pp. 49 - 81.

[6] Frédéric Planchet, Guillaume Serdeczny, *Modèle fréquence - coût : Quelles perspectives d'évolution ?*, 2014.

[7] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning*, Springer, 2015.

[8] SANJEEV KULKARNI, GILBERT HARMAN, *An Elementary Introduction to Statistical Learning Theory*, Wiley, 2011.

[9] François Husson, Sébastien Lê, Jérôme Pagès, *Exploratory Multivariate Analysis by Example Using R*, CRC Press, 2011.

[10] Andreas C. Müller and Sarah Guido, *Introduction to Machine Learning with Python*, O'Reilly Media, 2017.

[11] Darren Cook, *Practical Machine Learning with H2O*, O'Reilly Media, 2016.

[12] Ricco Rakotomalala, *Machine Learning with scikit-learn Python Programming.*

[13] MAITI Maher Maxim, *Modélisation de la prime pure en assurance automobile*, 2012.

[14] Jérôme CRÉTIEN, *Segmentation et provisionnement en réassurance non vie*, 2013.

[15] Jonathan KRIEGER, *L'exploraiton de données en assurance : apports de l'Analytics*, 2014.

[16] Erwan LE CORRE, *Constitution de groupes homogènes de risque dans le cadre de Solvabilité II*, 2012.

[17] Sébastien GUERIF, *Réduction de dimension en Apprentissage Numérique Non Supervisé*, 2006.

[18] Rémi BELLINA, *Méthodes d'apprentissage appliquées à la tarification non-vie*, 2014.

[19] Virginie POUNA SIEWE, *Modèles additifs généralisés : Intérêts de ces modèles en assurance automobile.*

[20] *Arbres binaires de décision*, github.com/wikistat.

[21] Padraic G. Neville, *Decision Trees for Predictive Modeling*, 1999. Terry M. Therneau, Elizabeth J. Atkinson, Mayo Foundation *An Introduction to Recursive Partitioning Using the RPART Routines*, March 12, 2017.

[22] Tianqi Chen, Carlos Guestrin, *XGBoost : A Scalable Tree Boosting System*, 2016.

[23] Tianqi Chen, *Introduction to Boosted Trees*, 2014.

[24] Tong He, *XGBoost eXtreme Gradient Boosting*, 2014.

[25] Tianqi Chen, Tong He, *Higgs Boson Discovery with Boosted Trees*, JMLR : Workshop and Conference Proceedings 42 :69-80, 2015.

[26] Hanspeter Pfister, Joe Blitzstein, and Verena Kaynig, *Ensemble Learning and Random Forests*, CS109/Stat121/AC209/E-109 Data Science.

[27] Sébastien Guérif, *Réduction de dimension en Apprentissage Numérique Non Supervisé*, 2006.

[28] G. E. Hinton and R. R. Salakhutdinov, *Reducing the Dimensionality of Data with Neural Networks*, DOI : 10.1126/science.1127647, Science 313, 504 (2006).

[29] G. E. Hinton, A. Krizhevsky, S. D. Wang, *Clustering algorithms and autoencoders for anomaly detection.*

[30] Chunfeng Song1, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan, *Auto-encoder Based Data Clustering.*

[31] Nikhil Ketkar, *Deep Learning with Python : A Hands-on Introduction*, Apress, 2017.

[32] Nikhil Buduma, Nicholas Lacascio, *Fundamentals of Deep Learning*, O'Reilly Media, 2017.

[33] Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, O'Reilly Media, 2017.

[34] Bernard Desgraupes, *Clustering Indices*, April 2013.

[35] Stéphane Lallich, Philippe Lenca, *Indices de qualité en clustering*, Journée Clustering 2015, 20 Octobre 2015, Orange Labs, Issy Les Moulineaux.

[36] Salem Alelyani, Jiliang Tang and Huan Liu, *Feature Selection for Clustering : A Review*

[37] Robert Tibshirani, Guenther Walther and Trevor Hastie, *Estimating the number of clusters in a data set via the gap statistic*, J.R.Statist.Soc.B (2001).

[38] André Bouchier, *Les classifications automatiques*, Montpellier, 2010.

[39] Emily Fox, Carlos Guestrin, *Clustering : Grouping Related Docs*, Machine Learning Specialization, University of Washington, 2016.

[40] Junyuan Xie, Ross Girshick, Ali Farhadi, *Unsupervised Deep Embedding for Clustering Analysis.*

[41] Charles Bouveyron, Camille Brunet, *Model-Based Clustering of High-Dimensional Data : A review*, Computational Statistics and Data Analysis, Elsevier, 2013, 71, pp.52-78

[42] Rocci Rakotomalala, *Classification automatique sous Python CAH et K-Means*, http ://eric.univ-lyon2.fr/ ricco/cours

[43] José Alfredo F. Costa, Márcio L. de Andrade Netto, *Clustering of complex shaped data sets via Kohonen maps and mathematical morphology* (2001).

[44] M.Cottrell, J.C.Fort, G.Pagès, *Theoretical Aspects of the SOM Algorithm*, 2017.

[45] Marie Cottrell, SmaïI Ibbou, Patrick Letrémy, Patrick Rousset, *Cartes auto-organisées pour l'analyse exploratoire de données et la visualisation.*

[46] Juha Vesanto and Esa Alhoniemi, *Clustering of the Self-Organizing Map*, IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 11, NO. 3, MAY 2000.

[47] Teuvo Kohonen, *MATLAB Implementations and Applications of the Self-Organizing Map*, Unigrafia Oy, Helsinki, Finland, 2014.

[48] Marie Cottrell et Patrick Letremy, *Algorithme de Kohonen : classification et analyse exploratoire des données.*

[49] Thomas Soubiran, *Cartes auto-organisatrices de Kohonen et typologisation de territoires*, Semin-R, 10 juin 2016.

[50] Juha Vesanto, *SOM-Based Data Visualization Methods*, Intelligent Data Analysis, Volume 3, Number 2, Elsevier Science, pp. 111-126, 1999.

[51] Ron Wehrens, Lutgarde M. C. Buydens, *Self- and Super-organizing Maps in R : The kohonen Package*, Journal of Statistical Software, October 2007, Volume 21, Issue 5.

[52] Volodymyr Melnykov, Ranjan Maitra, *Finite mixture models and model-based clustering*, Statistics Surveys Vol. 4 (2010) 80–116, ISSN : 1935-7516, DOI : 10.1214/09-SS053.

[53] Charles Bouveyron, Camille Brunet, *Model-Based Clustering of High-Dimensional Data : A review.* Computational Statistics and Data Analysis, Elsevier, 2013, 71, pp.52-78.

[54] Jean-Jacques Droesbeke,Gilbert Saporta,Christine Thomas-Agnan, *Modèles à variables latentes et modèles de mélanges.* Editions Techinip, Paris, 2013.