Institut des ACTUAIRES

Dauphine | PSL
UNIVERSITÉ PARIS

Mémoire présenté devant l'Université de Paris-Dauphine pour l'obtention du Certificat d'Actuaire de Paris-Dauphine et l'admission à l'Institut des Actuaires le 30/01/2023

Par : ***Adnane EL KASMI***

Titre : **Analyse comparative des méthodes de Machine Learning et de Deep Learning en provisionnement non-vie**

Confidentialité : ☑ Non ☐ Oui (Durée : ☐ 1 an ☐ 2 ans)

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

Membres présents du jury du Master Actuariat de Paris-Dauphine :

Louchard Michel

Jérôme Schaeffer

Quentin Guibert

Entreprise :
Nom : ***Addactis France***
Signature :

Directeur de Mémoire en entreprise :
Nom : ***Tuyen LE***
Signature :

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

Secrétariat :

Bibliothèque :

Signature du responsable entreprise

Signature du candidat

Résumé

Ces dernières années, de nouvelles méthodes basées sur l'apprentissage automatique et l'apprentissage profond en particulier ont révolutionné le travail des actuaires, y compris dans le domaine du provisionnement en assurance non-vie. Ces méthodes ont fait preuve d'obtenir une estimation centrale plus précise que celle obtenue par des méthodes usuelles comme Chain-Ladder et GLM ODP.

Cependant, à ce jour, les applications de ces méthodes dans le provisionnement en vision agrégée ont été principalement axées sur la prédiction de l'estimation centrale. Dans la pratique, l'estimation de la distribution des provisions est aussi importante pour l'allocation de capital qui satisfait aux exigences réglementaires, afin de mieux évaluer les risques de provisions et mieux connaître leurs portefeuilles de sinistres.

Ce mémoire applique les méthodes d'apprentissage automatique (arbre de décision, forêt aléatoire et XGBoost avec une régression tweedie) et le Mixture Density Network (MDN), un réseau de neurones qui se concentre sur la prévision probabiliste en appliquant un mélange gaussien aux données incrémentaux, pour obtenir simultanément une estimation centrale précise et un choix de distribution flexible. L'ajustement des modèles est effectué en utilisant une approche de type ROCV (Rolling-Origin Cross Validation).

Tout au long de ce mémoire, nous nous concentrons sur les triangles de développement (données incrémentaux) et nous montrons que les modèles MDN et XGBoost avec une régression tweedie sont plus performants que les méthodes usuelles, même avec des quantités de données relativement limitées. Nous utilisons à la fois des données simulées pour valider les résultats et des données réelles pour illustrer et vérifier le caractère pratique des approches.

Mots-clés : *provisonnement*, *méthodes usuelles*, *triangles de développement*, *Chain Ladder*, *Mack*, *Bornhuetter Ferguson*, *GLM ODP*, *boostrap*, *apprentissage automatique*, *apprentissage profond*, *arbre de décision*, *forêts aléatoires*, *XGBoost*, *réseaux de neurones*, *MDN*, *PSAP*.

Abstract

In recent years, new methods based on Machine Learning and Deep Learning in particular have revolutionized the work of actuaries, including in the field of non-life insurance reserving. These methods have proven to obtain a more accurate best estimate than that obtained by usual methods such as Chain-Ladder and GLM ODP.

However, to this day, the applications of these methods in aggregate reserving have been mainly focused on the prediction of the best estimate. In fact, estimating the distribution of reserves is also important for risk assessment and measurement, to better assess the risks of reserves and to better understand their claims portfolios.

This dissertation Applies Machine Learning methods (decision tree, random forest and XGBoost with tweedie regression) and the Mixture Density Network (MDN), a neural network that focuses on probabilistic forecasting by applying a mixture Gaussian to incremental data, to simultaneously obtain an accurate best estimate and a flexible distribution choice. Model fitting is performed using a Rolling-Origin Cross Validation (ROCV) approach.

Throughout this work, we focus on development triangles (incremental data) and show that MDN and XGBoost models with tweedie regression perform better than the usual methods, even with relatively limited amounts of data. We use both simulated data to validate the properties and real data to illustrate and verify the practicality of the approaches.

Keywords : *Reserving*, *usual methods*, *run off triangles*, *Chain Ladder*, *Mack*, *Bornhuetter Ferguson*, *GLM ODP*, *boostrap*, *machine learning*, *deep learning*, *decision tree*, *random forest*, *XGBoost*, *neural networks*, *MDN*, *P&C*.

Note de Synthèse

• **Contexte**

Le provisionnement en assurance non-vie est vital pour une compagnie d'assurance, tant pour la sécurité financière que pour la production de comptes de résultat précis. Les méthodes usuelles de provisionnement sont utilisées depuis de nombreuses années pour évaluer l'estimation centrale des assureurs. Le développement récent de l'intelligence artificielle, ainsi que l'amélioration incessante de la puissance de calcul des ordinateurs, ont permis de développer des modèles de Machine Learning et de Deep Learning dans le cadre du provisionnement non-vie, l'emploi de ces modèles est un sujet de recherche très actif en ce moment. Dans ce contexte, nous avons consacré notre mémoire à l'application des différents méthodes de Machine Learning et de Deep Learning, sur des triangles de développement, en utilisant une méthode innovante Rolling-Origin Cross-Validation, afin d'améliorer leur précision.

• **La méthode Rolling-Origin Cross-Validation (ROCV)**

Le triangle de développement peut être considéré comme un ensemble de séries temporelles, une pour chaque période d'accident. La méthode ROCV suppose une partition roulante d'apprentissage, de validation et de test, effectuée dans l'ordre chronologique. Plusieurs étapes sont effectuées afin d'améliorer l'efficacité dans la sélection de modèles.

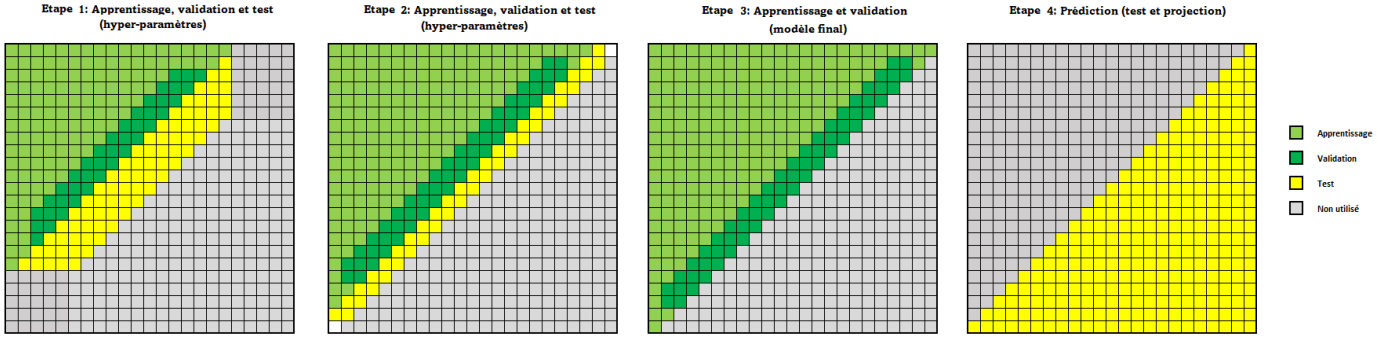

FIGURE 1 : Les 4 étapes du processus de modélisation.

▷ Dans la première étape, un grand échantillion des dernières périodes calendaires est utilisé pour l'ensemble de test. Cette partition se concentre sur l'évaluation de la précision de la prédiction à long terme du modèle.

▷ La deuxième étape laisse moins de périodes calendaires pour le test. Les périodes calendaires ultérieures sont incluses dans l'apprentissage, ce qui permet d'évaluer la capacité du modèle à capturer les tendances plus récentes et plus globales présentes dans le triangle.

▷ La troixième étape permettra l'apprentissage et la validation du modèle final, *après avoir sélectionner les meilleurs hyper-paramètres qui minimisent les erreurs dans l'étape 1 et l'étape 2 à la fois*, en utilisant *ROCV* avec *5-fold*.

▷ La quatrième étape permettra la projection et la prédiction du triangle inférieur.

- **Les modèles de ML et DL utilisés**

◦ **Arbre de décision (Decision Tree)**

Les arbres de décision sont des algorithmes de ML utilisées dans les problèmes de régression et de classification. Leur principe repose sur l'algorithme *CART* dans le but de construire une classification hiérarchique descendante des observations en des catégories homogènes par rapport à la variable à prédire, c'est-à-dire de trouver une partition qui sépare au mieux les différentes observations.

◦ **Forêt aléatoire (Random Forest)**

L'algorithme forêt aléatoire appartient à la famille des agrégations de modèles. Cet algorithme utilise des stratégies aléatoires "*bagging*" et s'appuie sur les arbres de décisions, notamment des arbres CART, en essayant de réduire au maximum la variance de ceux-ci. L'idée principale de cet algorithme est d'utiliser une agrégation d'un grand nombre de modèles tout en évitant le sur-apprentissage.

◦ **XGBoost (eXtreme Gradient Boosting)**

L'algorithme XGBoost appartient à la famille des méthodes ensemblistes qui bénéficent d'un traitement parallélisé optimisant les temps de calcul, en se basant sur la descente du gradient. Cette technique utilise un regroupement d'arbres qui vont s'entraîner sur des sous-parties différentes de la base d'apprentissage, puis faire voter ces sous-modèles afin de prendre la décision finale en faisant appel au *boosting*. Pour une meilleure performance, on utilisera la *régression Tweedie*.

◦ **Mixture Density Network (MDN)**

Le modèle MDN est un type de réseau de neurones artificiels, qui utilise l'hypothèse que toute distribution générale, peut être décomposée en un mélange de distributions normales, pour obtenir simultanément une estimation centrale précise et un choix de distribution flexible.

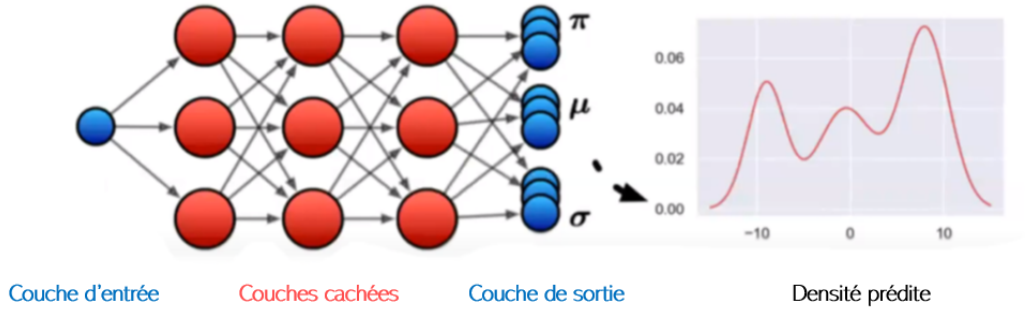

FIGURE 2 : Architecture du modèle Mixture Density Network.

Les montants incrémentaux $X_{i,j}$ sont supposées suivre une distribution mélange gaussienne

$$f_{\hat{X}_{i,j}}(x) = \sum_{k=1}^{K} \pi_{i,j,k} \phi\left(x \mid \mu_{i,j,k}, \sigma_{i,j,k}\right),$$

où ϕ est la fonction de densité de $\mathcal{N}(\mu, \sigma^2)$. D'après cette hypothèse de distribution, la couche de sortie du MDN estime les paramètres de la distribution du mélange, $(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\sigma})$, qui sont utilisés pour former une densité de mélange gaussien. Une fonction de perte de vraisemblance logarithmique négative (NLL) est utilisée

$$NLL\,\text{Loss}(\mathbf{X}, \hat{\mathbf{X}} \mid \mathbf{w}) = -\frac{1}{|\mathbf{X}|} \sum_{i,j:X_{i,j} \in \text{Apprentissage}} \ln\left(f_{\hat{X}_{i,j}}\left(X_{i,j} \mid \mathbf{w}\right)\right),$$

où $\mathbf{X}$ est l'ensemble des cellules $X_{i,j}$ dans l'ensemble d'apprentissage, $|\mathbf{X}|$ est le cardinal de $\mathbf{X}$, $\hat{\mathbf{X}}$ est l'ensemble des valeurs prédites de $X_{i,j}$ et $\mathbf{w}$ est l'ensemble des poids du MDN.

- **Présentation de données utilisées**
  - **Données simulées par SynthETIC (R)**

Grâce à la flexibilité accordée par SynthETIC (R) , quatre environnements ont été simulés, sous forme des triangles de charges incrémentales de taille 40 × 40 trimestres, aux caractéristiques et aux complexités différentes. Les environnements, étaient les suivants :

Environnement 1 - Branche de développement court : Cet environnement simule des sinistres à queue courte dont la composition est homogène pour tous les trimestres d'accident. Les charges de sinistres incrémentaux ayant un faible bruit et une homogénéité presque parfaite, ce qui permert aux méthodes usuelles de prédire l'estimation centrale quasi-parfaitement.

Environnement 2 - Variation de la cadence des sinistres : On simule un passage progressif d'une branche de développement long à une branche de développement cours. Au début, il y a plus de sinistres à long terme, mais la proportion de ces sinistres diminue, tandis que la proportion de sinistres à court terme augmente. Cet environnement introduit des complexités que les méthodes usuelles ne parviennent pas à saisir.

Environnement 3 - Branche de développement long avec choc d'inflation : L'inflation superposée passe instantanément de 0% à 10% par an, à partir du 30ème trimestre d'accident. Cet environnement teste la capacité des modèles à reconnaître les changements dans les effets calendaires et à adapter les projections en conséquence. Dans le triangle supérieur, seuls les 10 derniers trimestres civils contiennent des informations concernant le choc d'inflation, ce qui augmente la difficulté pour les modèles. Une autre complication est que la partition de ROCV contient peu de points d'apprentissage présentant la variation de l'inflation, d'où l'évaluation de sa capacité à saisir les tendances récentes.

Environnement 4 - Branche de développement long à forte volatilité : Cet environnement représente le triangle de developpement par défaut généré par le simulateur SynthETIC. La volatilité des sinistres est considérablement élevée. L'inflation superposée est de 30%, mais ce taux est multiplié par un facteur qui diminue à mesure que la charge de sinistre augmente. Ces tendances supplémentaires renforcent la volatilité qui était déjà présente.

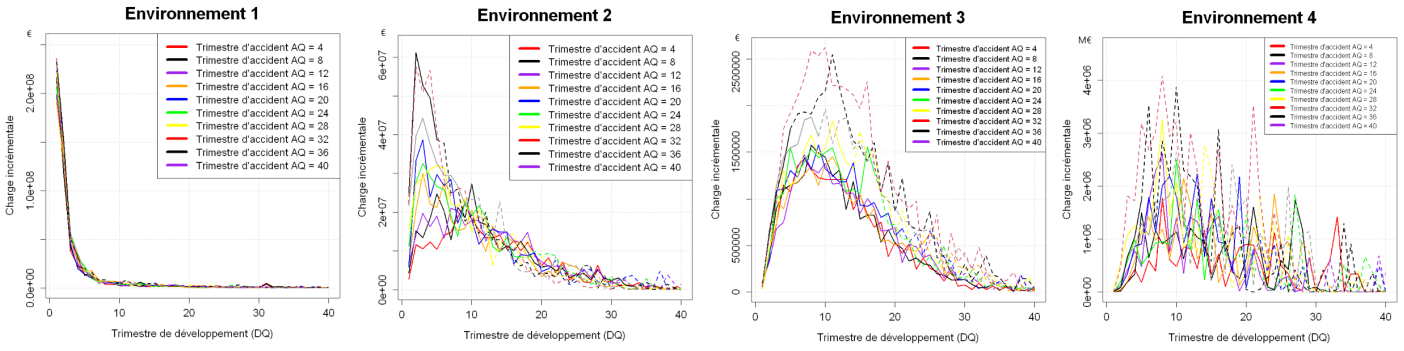

FIGURE 3 : Représentation des environnement simulées, les lignes pleines représentent les données du triangle supérieur, tandis que les lignes pointillées représentent les données du triangle inférieur.

◦ **Données réelles de ADDACTIS France**

Les données réelle sont issues des deux branches d'assurance non-vie, à savoir l'assurance automobile et l'assurance responsabilité civile générale. Les triangles de règlements incrémentaux utilisés sont de dimension **17x17** années (de 2004 à 2020). Les garanties, étaient les suivantes :

Automobile (RC matériel) : Cette garantie est obligatoire. Le montant de cette garantie est plafonné à 100M€ pour le matériel. Cette garantie permet de compenser financièrement les dommages matériels subis par les tiers lorsque la responsabilité de l'assuré est engagée à la suite d'accident ou dégats causés par le véhicule assuré.

Automobile (dommages) : Les biens garantis sont indemnisés selon le principe défini par l'article ***L121-1*** du ***code des assurances***. L'indemnité est égale au montant des réparations dans la limite de la valeur de remplacement du véhicule assuré.

Responsabilité civile générale : Cette garantie protège si une personne est tenu responsable d'avoir causé involontairement des dommages matériels dans notre cas à un tiers, soit par un produit ou par des activités d'exploitation.

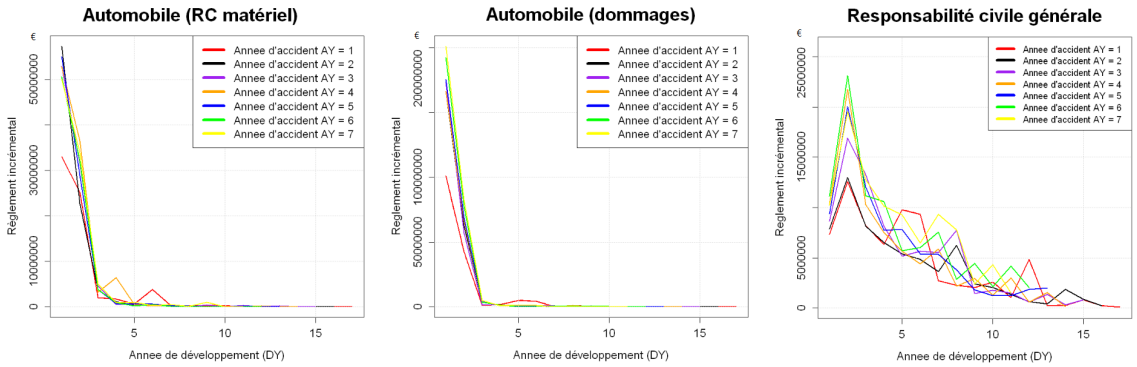

FIGURE 4 : Représentation des garanties des données réelles de ADDACTIS France.

• **Analyse et comparaison de résultats**

Les méthodes de Chain Ladder, Mack et GLM ODP donnent les même résultats pour l'estimation centrale, alors on se restreint au modèle GLM ODP qui va nous permettre aussi de comparer la distribution des provisions avec celle du MDN.

◦ **Données simulées par SynthETIC (R)**

◦◦ **L'estimation centrale**

Modèle	Provisions (M€)	Ratio (%)	RMSE (M€)	Temps de calcul
Simulée (réference)	1652.47	100.00	-	-
MDN	**1654.43**	**100.12**	1.13	6h 24min
XGBoost	1688.19	102.16	1.15	1h 42min
GLM ODP	1707.85	103.35	**1.12**	**instantané**
Forêt aléatoire	1722.95	104.27	1.19	14min 32sec
Arbre de décision	1621.52	98.13	1.21	2min 46sec

TABLE 1 : Provisions totales, Backtesting, score RMSE et temps de calcul pour l'environnement 1.

Pour l'environnement 1, les données satisfont par conception les hypothèses de GLM ODP, qui s'est donc avéré très compétitif. Néanmoins, les modèles MDN et XGBoost sont légèrement plus performants lorsqu'il s'agit de mesurer la précision des provisions totales. Cela peut être attribué à l'ajustement lisse du le MDN et à l'utilisation de la régression tweedie pour le XGBoost. Les modèles forêt aléatoire et arbre de décision donnent aussi des résultats satisfaisants.

Modèle	Provisions (M€)	Ratio (%)	RMSE (M€)	Temps de calcul
Simulée (réference)	3889.61	100.00	-	-
MDN	**4613.36**	**118.60**	**2.80**	7h 44min
XGBoost	4796.22	123.31	2.93	2h 2min
Arbre de décision	4815.85	123.81	3.21	3min 26sec
Forêt aléatoire	5418.03	139.29	3.66	1h 11min
GLM ODP	7905.54	203.25	8.63	**instantané**

Table 2 : Provisions totales, Backtesting, score RMSE et temps de calcul pour l'environnement 2.

Pour l'environnement 2, les modèles MDN et XGBoost ont appris avec succès que la vitesse de traitement des sinistres augmente, ce qui leur permet de prédire des provisions totales proches de celles simulées. Les modèles arbre de décision et forêt aléatoire donnent des résultats mois précis. Le modèle GLM ODP, en supposant l'homogénéité de l'évolution des sinistres, a approximé les sinistres comme étant à queue moyenne, ce qui a conduit à une sur-estimation remarquable des sinistres dans les derniers trimestes d'accident.

Modèle	Provisions (M€)	Ratio (%)	RMSE (M€)	Temps de calcul
Simulée (réference)	435.65	100.00	-	-
MDN	**476.14**	**109.29**	**0.185**	7h 58min
XGBoost	384.12	88.17	0.214	4h 9min
Forêt aléatoire	359.36	82.49	0.227	37min 43sec
Arbre de décision	353.05	81.04	0.231	27sec
GLM ODP	337.32	77.43	0.241	**instantané**

Table 3 : Provisions totales, Backtesting, score RMSE et temps de calcul pour l'environnement 3.

Pour l'environnement 3, les modèles MDN et XGBoost ont capturé le choc d'inflation à partir du 20ème trimestre calendaire. Le GLM ODP n'a pas suivi le rythme de l'augmentation de l'inflation en raison de sa capacité limitée à gérer l'hétérogénéité, ce qui a conduit à une sous-estimation des provisions totales. Les modèles arbre de décision et forêt aléatoire donnent des résultats moyens.

Modèle	Provisions (M€)	Ratio (%)	RMSE (M€)	Temps de calcul
Simulée (réference)	463.78	100.00	-	-
MDN	**497.70**	**107.31**	**0.582**	7h 52min
XGBoost	536.10	115.59	0.601	1h 2min
GLM ODP	550.55	118.71	0.741	**instantané**
Forêt aléatoire	572.26	123.39	0.658	36min 52sec
Arbre de décision	629.25	135.68	0.640	3min 25sec

Table 4 : Provisions totales, Backtesting, score RMSE et temps de calcul pour l'environnement 4.

Pour l'environnements 4, les modèles MDN et XGBoost ont bien traité les données volatiles et ont fourni des estimations centrales précises, surpassant les autres modèles.

Enfin, les modèles MDN et XGBoost ont produit d'excellentes prévisions d'estimation centrale dans tous les environnements. Lorsque l'environnement présentait une plus grande hétérogénéité structu-

relle, c'est-à-dire lorsque les hypothèses de Chain-ladder ne sont pas satisfaites, les modèles MDN et XGBoost ont largement dépassé le modèle GLM ODP en termes de précision.

○○ La densité des provisions totales

Les distributions du GLM ODP sont obtenues par la méthode *Boostrap*. Les densités simulées des environnements sont obtenues en simulant *200 triangles* du même environnement.

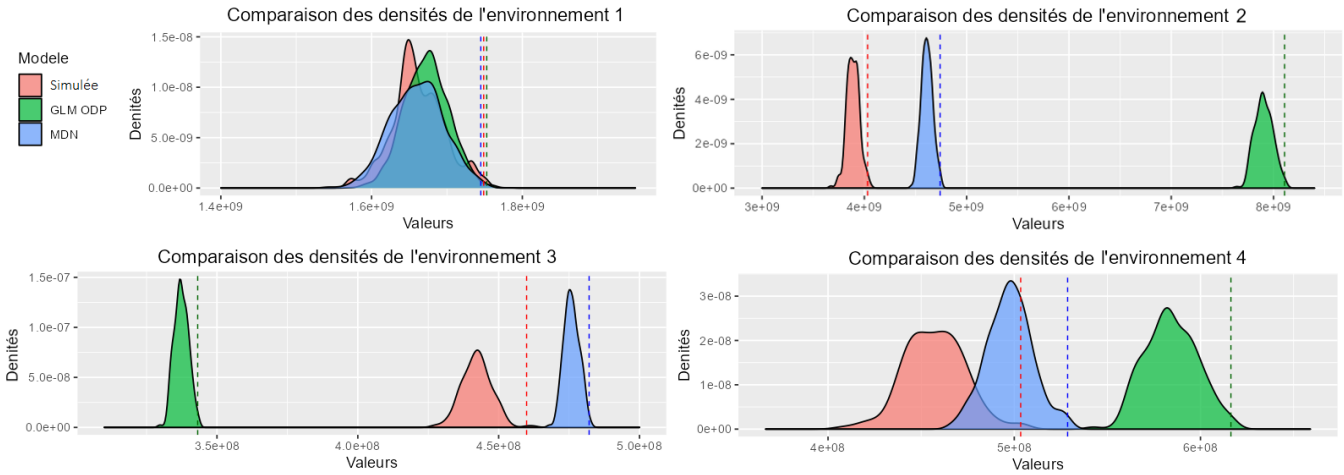

FIGURE 5 : Comparaison des densités de provisions totales pour les 4 environnements.

Le modèle MDN a révélé des estimations centrales et des répartitions des provisions totales plus précises que celles du GLM ODP. La précision de la distribution du MDN est due à sa capacité à modéliser des tendances complexes, ainsi qu'à sa capacité à ajuster une distribution plus flexible.

Environnement	Modèle	SCR_{Ultime} (M€)	CoV (%)
1	Simulée	96.27	21.11
1	MDN	90.71	21.57
1	GLM ODP	44.93	17.74
2	Simulée	142.47	15.47
2	MDN	127.41	12.69
2	GLM ODP	201.92	11.55
3	Simulée	24.27	13.87
3	MDN	5.96	9.94
3	GLM ODP	5.83	8.71
4	Simulée	39.68	38.58
4	MDN	30.97	24.42
4	GLM ODP	65.80	24.55

TABLE 5 : Comparaison des SCR_{Ultime} et des CoV pour les 4 environnements.

Pour tous les environnements, le modèle MDN a fourni des estimations de SCR_{Ultime} et de CoV précises et très proches de celles simulées, surpassant celles du modèle GLM ODP.

○ Données réelles de ADDACTIS France

Les résultats obtenus pour les données réelles sont comparés avec celles de l'expert (réalisé par une équipe d'expert), qui possède l'historique des règlements depuis 1990 et qui utilise un modèle Mack avec des retraitements (l'exclusion des facteurs de développement non pertinents et l'utilisation des facteurs de queues).

○○ L'estimation centrale

Modèle	Provisions (M€)	Ratio (%)	RMSE (M€)	Temps de calcul
Expert (réference)	43.35	100.00	-	-
MDN	**44.08**	**102.15**	1.06	4h 19min
XGBoost	41.98	97.08	**0.74**	26min
Forêt aléatoire	41.63	96.03	1.64	3min 47sec
Arbre de décision	45.62	105.24	1.63	1min 36sec
GLM ODP	36.84	85.13	1.76	**instantané**

TABLE 6 : Provisions totales, Backtesting, score RMSE et temps de calcul pour la garantie automobile (RC matériel).

La garantie automobile (RC matériel) est similaire à l'environnement 1. Cette garantie contient des règlements à queue courte dont la composition est quasi-homogène pour tous les années d'accident. Les modèles de Machine Learning et de Deep Learning sont légèrement plus performants que le modèles GLM ODP en termes de précision des provisions totales. Cela peut être attribué à l'ajustement lisse du le MDN, à l'utilisation de la régression tweedie pour le XGBoost et à la méthode ROCV, qui a permis de sélectionner les meilleurs modèles possibles.

Modèle	Provisions (M€)	Ratio (%)	RMSE (M€)	Temps de calcul
Expert (réference)	70.28	100.00	-	-
MDN	**71.20**	**99.88**	1, 73	4h 27min
XGBoost	70.07	99.70	**1, 65**	27min
Forêt aléatoire	72.39	103.01	3, 36	3min 55sec
GLM ODP	67.11	95.49	4, 67	**instantané**
Arbre de décision	75.61	107.59	2, 12	1min 43sec

TABLE 7 : Provisions totales, Backtesting, RMSE et temps de calcul pour automobile (dommages).

La garantie automobile (dommages) est ressemblable à l'environnement 1, mais avec plus de volatilité. Les modèles MDN, XGBoost et forêt aléatoire se sont bien adaptés avec ces volatilités. Par contre, les modèles GLM ODP et arbre de décision n'ont pas pu suivre le rythme de cette volatilité, ce qui a conduit à une sous-estimation des provisions totales.

Modèle	Provisions (M€)	Ratio (%)	RMSE (M€)	Temps de calcul
Expert	439.64	100.00	-	-
MDN	**435.28**	**99.01**	**1, 33**	4h 13min
Forêt aléatoire	434.59	98.85	1, 56	3min 59sec
XGBoost	400.21	91.03	1, 46	25min
Arbre de décision	347.16	78.96	2, 73	1min 57sec
GLM ODP	322.56	73.37	3, 63	**instantané**

TABLE 8 : Provisions totales, Backtesting, RMSE et temps de calcul pour responsabilité civile générale.

Pour la garantie responsabilité civile générale, les modèles MDN, forêt aléatoire et XGBoost ont bien traité les données volatiles et ont fourni des estimations centrales précises, surpassant les autres modèles.

Enfin, les modèles MDN et XGBoost ont produit d'excellentes prévisions d'estimation centrale dans tous les environnements. Lorsque l'environnement présentait une plus grande hétérogénéité structurelle, c'est-à-dire lorsque les hypothèses de Chain-ladder ne sont pas satisfaites, les modèles MDN et XGBoost ont largement dépassé le modèle GLM ODP en termes de précision.

○○ La densité des provisions totales

Les distributions du GLM ODP sont obtenues par la méthode *Boostrap*. Les densités de l'expert sont obtenues en appliquant un *Boostrap* au modèle Mack.

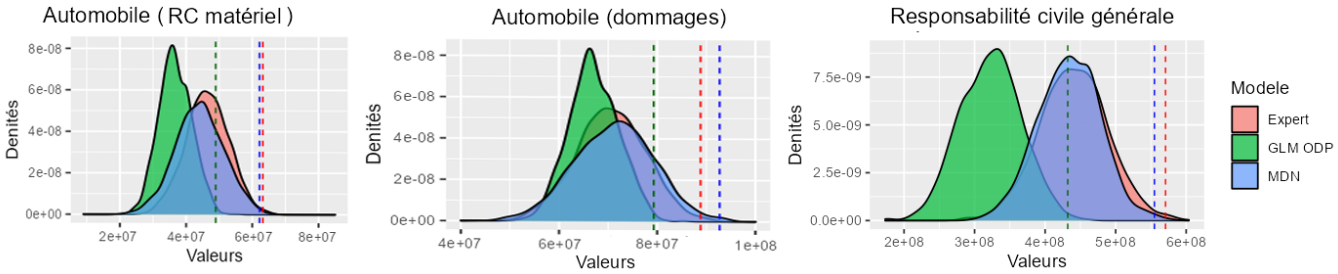

FIGURE 6 : Comparaison des densités de provisions totales pour les 3 garanties.

Le modèle MDN a révélé des estimations centrales et des répartitions des provisions totales plus précises que celles du GLM ODP. La précision de la distribution du MDN est due à sa capacité à modéliser des tendances complexes, ainsi qu'à sa capacité à ajuster une distribution plus flexible.

Garantie	Modèle	SCR_{Ultime} (M€)	CoV (%)
Automobile (RC matériel)	Expert	19.84	14.08
Automobile (RC matériel)	MDN	18.10	16.67
Automobile (RC matériel)	GLM ODP	12.09	13.54
Automobile (dommages)	Expert	18.24	10.65
Automobile (dommages)	MDN	21.44	11.38
Automobile (dommages)	GLM ODP	12.23	7.52
Responsabilité civile générale	Expert	130.02	10.65
Responsabilité civile générale	MDN	120.77	10.21
Responsabilité civile générale	GLM ODP	109.22	11.00

TABLE 9 : Comparaison des SCR_{Ultime} et des CoV pour les 3 garanties.

Pour tous les garanties, le modèle MDN a fourni des estimations de SCR_{Ultime} et de CoV précises et très proches de celles de l'expert, surpassant celles du modèle GLM ODP.

• Les limites

Malgré les perfomances remarquable des méthodes de Machine Learning et de Deep Learning, ces méthodes rencontrent plusieurs limites. Par exemple, les modèles nécessitent souvent un temps plus important que celui des méthodes usuelles pour avoir les résultats. En outre, ces modèles sont exposés au manque d'interprétabilité et la capacité d'expliquer, de piloter ou de présenter des informations dans des termes humainement compréhensibles.

Cependant, quelques modèles requièrent que les montants incrémentaux $X_{i,j}$ soint positifs, tandis que les montants incrémentaux $X_{i,j}$ sont souvent négatifs à cause de la présence de boni de liquidation dans le déroulement de la charge des sinistres, ou l'encaissement de recours en fin de développement, pour les triangles de règlements.

Les modèles de Machine Learning et Deep Learning avec la méthode ROCV ouvrent des perspectives d'automatisation et pourraient, par exemple, être utile dans le cadre de processus de revue indépendante de provisionnement.

Synthesis note

- **Context**

The provisioning in non-life insurance is vital for an insurance company, both for financial security and for the production of accurate income statements. The usual provisioning methods have been used for many years to assess the central estimate of insurers. The recent expansion of artificial intelligence, as well as the constant improvement of computers' computing power, have made it possible to develop Machine Learning and Deep Learning models in the context of non-life provisioning, the use of these models is at the moment, a very active topic of research. Within this context, we have devoted our thesis to the application of different Machine Learning and Deep Learning methods, on development triangles, using an innovative Rolling-Origin Cross-Validation method, in order to improve their accuracy.

- **Rolling-Origin Cross-Validation (ROCV) Method**

The development triangle can be thought of as a set of time series, one for each accident period. The ROCV method assumes a rolling score of training, validation, and testing, performed in chronological order. Several steps are performed in order to improve efficiency in selection of models.

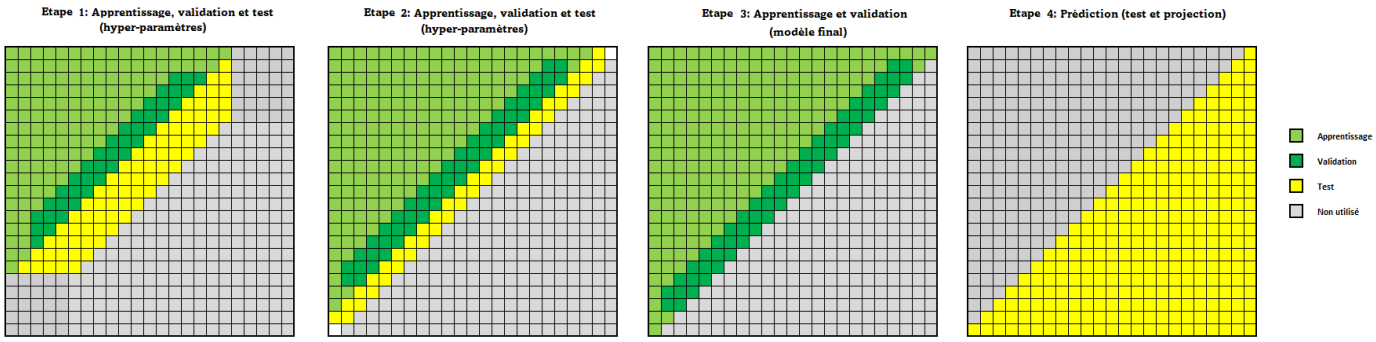

Figure 7: The 4 steps of the modeling process.

▷ In the first step, a large sample of the last periods are used for all tests. This partition focuses on evaluating the long-term prediction accuracy of the model.

▷ The second step leaves less calendar periods for the test. Later calendar periods are included in the training, which helps assess the model's ability to capture more recent and more global trends present in the triangle.

▷ The third step will allow the training and validation of the final model, *after selecting the best hyper-parameters that minimize errors in both step 1 and step 2*, using *ROCV* with *5-fold*.

▷ The fourth step will allow the projection and prediction of the lower triangle.

- **ML and DL models used**

  - **Decision Tree (Arbre de décision)**

Decision trees are ML algorithms used in regression and classification problems. Their principle is based on the algorithm *CART* with the aim of constructing a descending hierarchical classification of observations into homogeneous categories with respect to the variable to be predicted, i.e. to find a partition which best separates the various observations.

  - **Random Forest (Forêt aléatoire)**

The Random Forest algorithm belongs to the family of model aggregations. This algorithm uses "*bagging*" random strategies and is based on decision trees, in particular CART trees, while trying to reduce their variance as much as possible. The main idea of this algorithm is to use an aggregation of a large number of models while avoiding overlearning.

  - **XGBoost (eXtreme Gradient Boosting)**

The XGBoost algorithm belongs to the family of ensemble methods that benefit from parallelized processing that optimizes computation times, based on the descent of the gradient. This technique uses a grouping of trees which will train on different sub-parts of the learning base, then, have these sub-models voted on in order to make the final decision by using *boosting*. For better performance, we will use the *Tweedie regression*.

  - **Mixture Density Network (MDN)**

The MDN model is a type of artificial neural network, which uses the assumption that any general distribution can be decomposed into a mixture of normal distributions to simultaneously achieve accurate central estimation and flexible distribution choice.

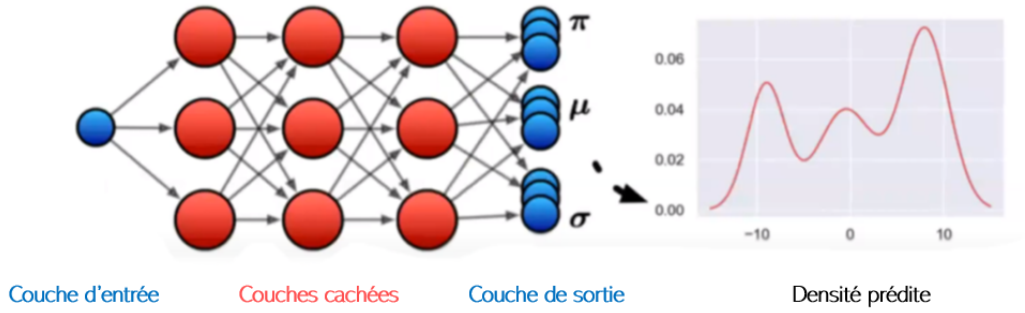

Figure 8: Architecture of the model Mixture Density Network.

The incremental charges (or payments) $X_{i,j}$ are assumed to follow a Gaussian mixture distribution

$$f_{\hat{X}_{i,j}}(x) = \sum_{k=1}^{K} \pi_{i,j,k} \phi\left(x \mid \mu_{i,j,k}, \sigma_{i,j,k}\right),$$

is the density function of $\mathcal{N}(\mu, \sigma^2)$. This distribution assumption, the MDN output layer estimates the parameters of the mixture distribution, $(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\sigma})$, which are used to form a Gaussian mixing density. A negative logarithmic loss-of-likelihood function (LLR)

$$NLL\,\text{Loss}(\mathbf{X}, \hat{\mathbf{X}} \mid \mathbf{w}) = -\frac{1}{|\mathbf{X}|} \sum_{i,j:X_{i,j} \in \text{Train}} \ln\left(f_{\hat{X}_{i,j}}\left(X_{i,j} \mid \mathbf{w}\right)\right),$$

where $\mathbf{X}$ is the set of cells $X_{i,j}$ in the training set, $|\mathbf{X}|$ is the cardinality of $\mathbf{X}$, $\hat{\mathbf{X}}$ is the set of predicted distributions of $X_{i,j}$ and $\mathbf{w}$ is the set of MDN weights.

- **Presentation of used data**
  - **Simulated data by SynthETIC (R)**

Thanks to the flexibility granted by SynthETIC (R), four environments were simulated, in the form of of incremental loads triangles of size 40 $\times$ 40 quarters, with different characteristics and complexities. The environments were as follows:

Environment 1 - Short development branch: This environment simulates short-tail claims whose composition is homogeneous for all accident quarters. Incremental claims charges have low noise and near-perfect homogeneity, allowing standard methods to predict the central estimate near-perfectly.

Environment 2 - Variation in the rate of claims: A gradual transition from a long-term development branch to a short-term development branch is simulated. At the beginning, there are more long-term claims, but the proportion of these claims decreases, while the proportion of short-term claims increases. This environment introduces complexities that the usual methods fail to capture.

Environment 3 - Long development branch with inflation shock: Superimposed inflation goes instantly from 0% to 10% per year, starting from the 30th quarter of the accident. This environment tests the ability of models to recognize changes in calendar effects and adapt projections accordingly. In the upper triangle, only the last 10 calendar quarters contain information regarding the inflation shock, which makes it more difficult for the models. Another complication is that the ROCV partition contains few learning points showing the variation in inflation, hence the assessment of its ability to capture recent trends.

Environment 4 - High volatility long development branch: This environment represents the default development triangle generated by the SynthETIC simulator. Claims volatility is incredibly high. The superimposed inflation is 30%, but this rate is multiplied by a factor which decreases as the loss load increases. These additional trends add to the volatility that was already there.

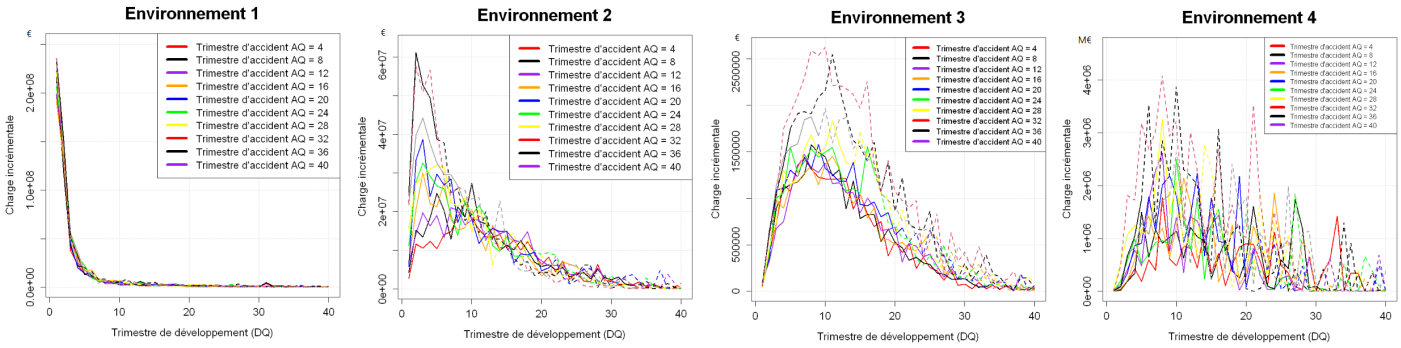

Figure 9: Representation of the simulated environments, the solid lines represent the upper triangle data, while the dotted lines represent the lower triangle data.

◦ **Real data from ADDACTIS France**

Real data comes from the two branches of non-life insurance, specifically motor insurance and general liability insurance. The incremental settlement triangles used are of dimension *17x17* years (from 2004 to 2020). The guarantees were as follows:

Motor Third Party Liability Insurance (MPTL) material damages : This guarantee is mandatory. The amount of this guarantee is capped at €100 million for equipment. This cover provides financial compensation for material damage by third parties when responsibility of the insured is incurred following an accident or damage caused by the insured vehicle.

Motor damage : The guaranteed goods are compensated according to the principle defined by article ***L121-1*** of the ***Insurance Code***. The indemnity is equal to the amount of the repairs within a limit of the replacement value of the insured vehicle.

General Liability : This warranty protects if a person is held liable for unintentionally causing property damage in our case to a third party, either by a product or by operating activities.d'exploitation.

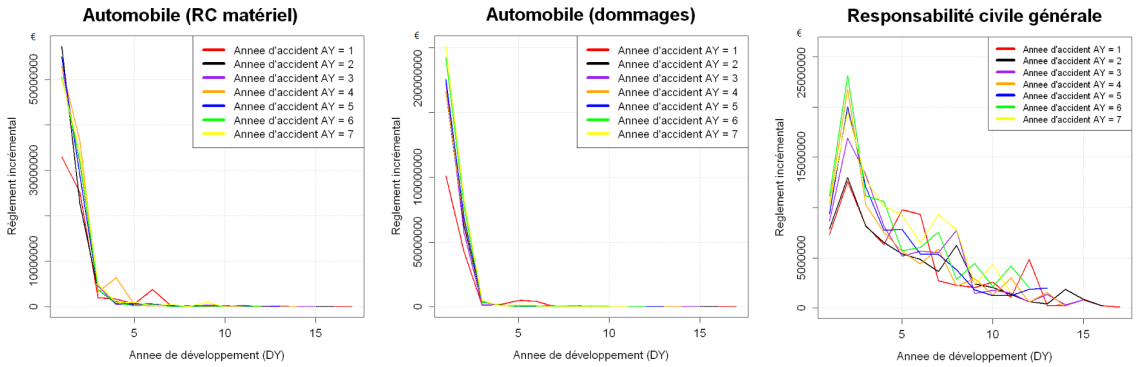

Figure 10: Representation of the guarantees of the real data of ADDACTIS France.

• **Analysis and comparison of results**

The methods of Chain Ladder, Mack and GLM ODP give the same results for the estimation of the Best Estimate, so we restrict ourselves to the GLM ODP model which also allows us to compare the distribution of provisions with the cell of the MDN.

◦ **Data simulated by SynthETIC (R)**

◦◦ **Best estimate**

Model	Total reserves (M€)	Ratio (%)	RMSE (M€)	Calculation time
Simulated (reference)	1652.47	100.00	-	-
MDN	**1654.43**	**100.12**	1.13	6h 24min
XGBoost	1688.19	102.16	1.15	1h 42min
GLM ODP	1707.85	103.35	**1.12**	**instant**
Random Forest	1722.95	104.27	1.19	14min 32sec
Decision Tree	1621.52	98.13	1.21	2min 46sec

Table 10: Total reserves, Backtesting, RMSE score and calculation time for the environment 1.

For environment 1, the data satisfies by design the assumptions of GLM ODP, which therefore turned out to be very competitive. Nevertheless, the MDN and XGBoost models perform slightly better when it comes to measuring the accuracy of total provisions. This can be attributed to the smooth fit of the MDN and the use of tweedie regression for the XGBoost. The Random Forest and Decision Tree models also give satisfactory results.

Model	Total reserves (M€)	Ratio (%)	RMSE (M€)	Calculation time
Simulated (reference)	3889.61	100.00	-	-
MDN	**4613.36**	**118.60**	**2.80**	7h 44min
XGBoost	4796.22	123.31	2.93	2h 2min
Decision Tree	4815.85	123.81	3.21	3min 26sec
Random Forest	5418.03	139.29	3.66	1h 11min
GLM ODP	7905.54	203.25	8.63	**instant**

Table 11: Total reserves, Backtesting, RMSE score and calculation time for the environment 2.

For Environment 2, the MDN and XGBoost models have successfully learned that claims processing speed increases, allowing them to predict total reserves close to the simulated ones. The Very Decision and Random Forest models give less accurate results. The GLM ODP model, assuming homogeneity in claims development, approximated claims as being medium-tailed, which led to a remarkable overstatement of claims in the later accident quarters.

Model	Total reserves (M€)	Ratio (%)	RMSE (M€)	Calculation time
Simulated (reference)	435.65	100.00	-	-
MDN	**476.14**	**109.29**	**0.185**	7h 58min
XGBoost	384.12	88.17	0.214	4h 9min
Random Forest	359.36	82.49	0.227	37min 43sec
Decision Tree	353.05	81.04	0.231	27sec
GLM ODP	337.32	77.43	0.241	**instant**

Table 12: Total reserves, Backtesting, RMSE score and calculation time for the environment 3.

For environment 3, the MDN and XGBoost models captured the inflation shock from the 20th calendar quarter. The GLM ODP has not kept pace with rising inflation due to its limited ability to handle heterogeneity, leading to an understatement of total provisions. The Decision Very and Random Forest models give average results.

Model	Total reserves (M€)	Ratio (%)	RMSE (M€)	Calculation time
Simulated (reference)	463.78	100.00	-	-
MDN	**497.70**	**107.31**	**0.582**	7h 52min
XGBoost	536.10	115.59	0.601	1h 2min
GLM ODP	550.55	118.71	0.741	**instant**
Random Forest	572.26	123.39	0.658	36min 52sec
Decision Tree	629.25	135.68	0.640	3min 25sec

Table 13: Total reserves, Backtesting, RMSE score and calculation time for the environment 4.

For Environment 4, the MDN and XGBoost models handled the volatile data well and provided accurate central estimates, outperforming the other models.

Finally, the MDN and XGBoost models produced excellent central estimate forecasts in all environments. When the environment exhibited greater structural heterogeneity, i.e. when the Chain-ladder

assumptions are not satisfied, the MDN and XGBoost models significantly outperformed the GLM ODP model in terms of accuracy.

○○ Density of total reserves

The GLM ODP distributions are obtained by the *Boostrap* method. The simulated densities of the environments are obtained by simulating *200 triangles* of the same environment.

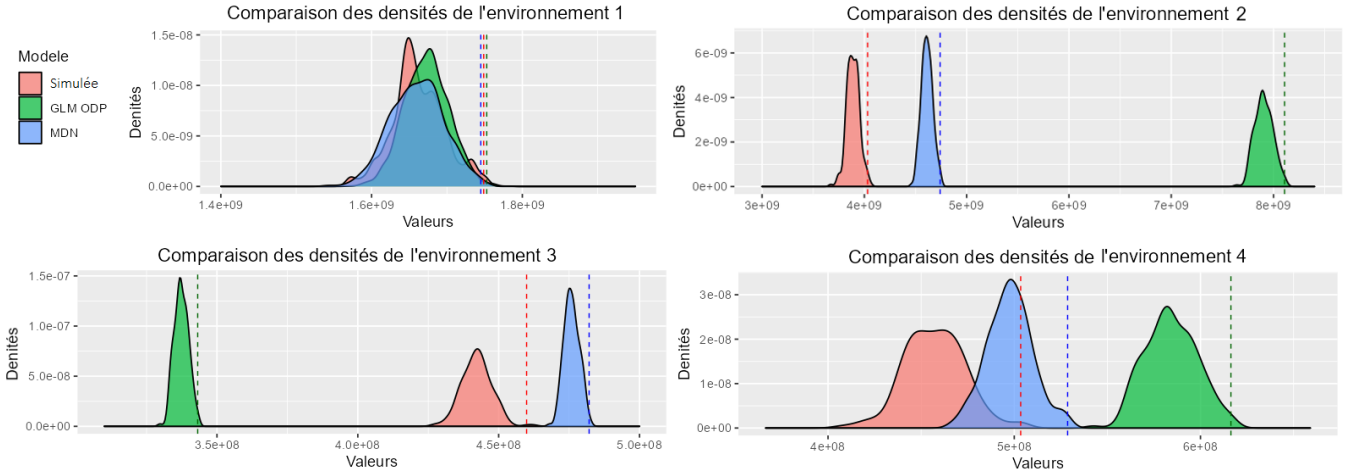

Figure 11: Comparison of total reserves densities for the 4 environments.

The MDN model revealed more accurate central estimates and allocations of total provisions than those of the GLM ODP. The accuracy of the MDN distribution is due to its ability to model complex trends, as well as its ability to fit a more flexible distribution.

Environnement	Model	SCR_{Ultime} (M€)	CoV (%)
1	Simulated	96.27	21.11
1	MDN	90.71	21.57
1	GLM ODP	44.93	17.74
2	Simulated	142.47	15.47
2	MDN	127.41	12.69
2	GLM ODP	201.92	11.55
3	Simulated	24.27	13.87
3	MDN	5.96	9.94
3	GLM ODP	5.83	8.71
4	Simulated	39.68	38.58
4	MDN	30.97	24.42
4	GLM ODP	65.80	24.55

Table 14: Comparison of SCR_{Ultime} and CoV for the 4 environments.

For all environments, the MDN model provided accurate $SCR_{Ultimate}$ and CoV estimates very close to the simulated ones, outperforming those of the GLM ODP model.

○ Real data from ADDACTIS France

The results obtained for the real data are compared with those of the expert, who has the history of the regulations since 1990 and who uses a Mack model with restatements.

○○ **Best estimate**

Model	Total reserves (M€)	Ratio (%)	RMSE (M€)	Calculation time
Expert (reference)	43.35	100.00	-	-
MDN	**44.08**	**102.15**	1.06	4h 19min
XGBoost	41.98	97.08	**0.74**	26min
Random Forest	41.63	96.03	1.64	3min 47sec
Decision Tree	45.62	105.24	1.63	1min 36sec
GLM ODP	36.84	85.13	1.76	**instant**

Table 15: Total reserves, Backtesting, RMSE score and calculation time for automobile (CL equipment).

MPTL material guarante is similar to environment 1. This guarantee contains short-tail regulations whose composition is nearly homogeneous for all accident years. The Machine Learning and Deep Learning models perform slightly better than the GLM ODP models in terms of accuracy of total provisions. This can be attributed to the smooth fit of the MDN, the use of tweedie regression for the XGBoost, and the ROCV method, which helped select the best possible models.

Model	Total reserves (M€)	Ratio (%)	RMSE (M€)	Calculation time
Expert (reference)	70.28	100.00	-	-
MDN	**71.20**	**99.88**	1,73	4h 27min
XGBoost	70.07	99.70	**1,65**	27min
Random Forest	72.39	103.01	3,36	3min 55sec
GLM ODP	67.11	95.49	4,67	**instant**
Decision Tree	75.61	107.59	2,12	1min 43sec

Table 16: Total reserves, Backtesting, RMSE score and calculation time for automobile (damage).

The guarante Motor damage is similar to environment 1, but with more volatility. The MDN, XGBoost and Random Forest models adapted well with these volatilities. On the other hand, the GLM ODP and Decision Tree models could not keep pace with this volatility, which led to an underestimation of the total provisions.

Model	Total reserves (M€)	Ratio (%)	RMSE (M€)	Calculation time
Expert (reference)	439.64	100.00	-	-
MDN	**435.28**	**99.01**	**1,33**	4h 13min
Random Forest	434.59	98.85	1,56	3min 59sec
XGBoost	400.21	91.03	1,46	25min
Decision Tree	347.16	78.96	2,73	1min 57sec
GLM ODP	322.56	73.37	3,63	**instant**

Table 17: Total reserves, Backtesting, RMSE score and calculation time for General Civil Liability).

For general liability, the MDN, Random Forest, and XGBoost models handled volatile data well and provided accurate central estimates, outperforming the other models.

Finally, the MDN and XGBoost models produced excellent central estimate forecasts in all environments. When the environment exhibited greater structural heterogeneity, i.e. when the Chain-ladder assumptions are not satisfied, the MDN and XGBoost models significantly outperformed the GLM ODP model in terms of accuracy.

○○ Density of total reserves

The GLM ODP distributions are obtained by the *Boostrap* method. Expert densities are obtained by applying a *Boostrap* to the Mack model.

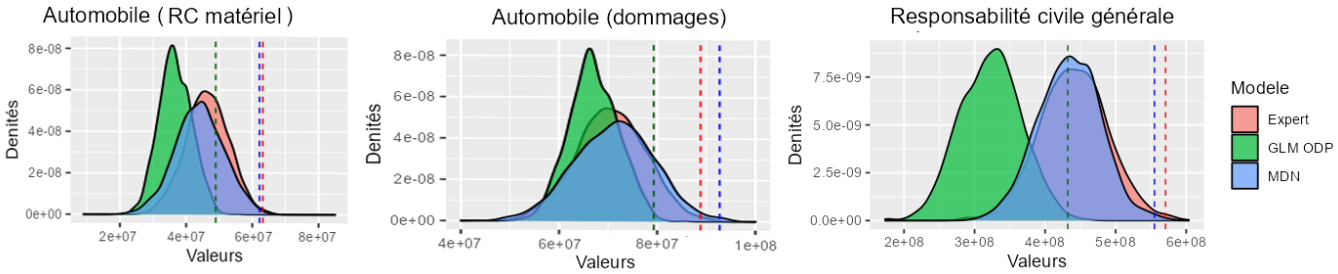

Figure 12: Comparison of total reserves densities for the 3 guarantees.

The MDN model revealed more accurate central estimates and allocations of total provisions than those of the GLM ODP. The accuracy of the MDN distribution is due to its ability to model complex trends, as well as its ability to fit a more flexible distribution.

Garantie	Model	SCR_{Ultime} (M€)	CoV (%)
MPTL	Expert	19.84	14.08
MPTL	MDN	18.10	16.67
MPTL	GLM ODP	12.09	13.54
Motor damage	Expert	18.24	10.65
Motor damage	MDN	21.44	11.38
Motor damage	GLM ODP	12.23	7.52
General Liability	Expert	130.02	10.65
General Liability	MDN	120.77	10.21
General Liability	GLM ODP	109.22	11.00

Table 18: Comparaison des SCR_{Ultime} et des CoV pour les 3 garanties.

For all guarantees, the MDN model provided estimates of $SCR_{Ultimate}$ and CoV that were accurate and very close to those of the expert, surpassing those of the GLM ODP model.

• The limits

Despite the remarkable performance of Machine Learning and Deep Learning methods, these methods have several limitations. For example, models often require more time than usual methods to get the results. Furthermore, these models are prone to lack of interpretability and the ability to explain, drive or present information in humanly understandable terms.

However, some models require the incremental amounts $X_{i,j}$ to be positive, while the incremental amounts $X_{i,j}$ are often negative due to the presence of liquidation bonuses in the course of the settlement. handling of claims, or collection of appeals at the end of development, for settlement triangles.

Machine Learning and Deep Learning models with the ROCV method open up prospects for automation and could, for example, be useful in the context of independent provisioning review processes.

Remerciements

J'adresse un remerciement très spécial à toutes les personnes qui m'ont aidé à faire aboutir ce mémoire.

Je voudrais dans un premier temps remercier, ma tutrice de stage Mme. *Tuyen LE*, manager au sein du cabinet Addactis, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

Je remercie également *Seifallah MALEK*, consultant actuaire chez Addactis, qui m'a aidé à comprendre davantage la réglementation solvabilité 2, la norme IFRS 17 et de mener à bien mon travail dans les délais fixés.

Je tiens à exprimer toute ma reconnaissance à tous les membres de l'équipe Modeling & Risk P&C pour leur contribution et leur accueil pendant mon stage de fin d'étude, particulièrement *Benjamin POUDRET*, responsable de l'équipe et *Romain BOYER-CHAMMARD*, senior manager, pour la confiance qu'ils m'ont accordée à travers différentes missions et leurs précieux conseils pour l'interprétation des résultats.

Merci également à M. *Franck KOFFI*, M. *Auriol WABO*, Mme *Nathalie BEDI*, M. *Romain NOBIS* et Mme *Isabelle JOULIN* pour leur soutien tout au long de ce mémoire.

Je tiens à remercier aussi M. *Bryan GAUTIER* et M. *Nabil RACHDI* pour la relecture scrupuleuse du manuscrit et leurs suggestions toujours avisées et surtout pour leurs lumières sur les sujets Machine Learning et Deep Learning qui m'ont beaucoup inspiré tout au long de mes travaux.

Ce mémoire doit également énormément à M. *Christophe DUTANG*, mon tuteur Dauphine, pour son suivi, son assistance précieuse et ses remarques pertinentes aux différents moments de mon mémoire.

Avec beaucoup d'égard, je remercie aussi l'ensemble du corps professoral et administratif de l'université Paris Dauphine et l'ISUP pour la qualité de l'enseignement dispensé et leurs accompagnements.

Enfin, mes remerciements vont au personnel d'ADDACTIS qui m'a chaleureusement accueilli au sein de leur structure.

Ces remerciements ne peuvent s'achever, sans une pensée pour *mes très chers parents*. Leur présence et leur soutien inconditionnel sont pour moi les piliers fondateurs de ce que je suis et de ce que je fais.

Table des matières

Introduction

Le provisionnement en assurance non-vie est vital pour une compagnie d'assurance, tant pour la sécurité financière que pour la production de comptes de résultat précis. Bien entendu, il est extrêmement improbable que la provision pour sinistre à payer soit un jour le montant exact nécessaire pour régler tous les sinistres pour lesquels elle est prévue, parce que les provisions totales ne peuvent être connues qu'après des années. Il existe plusieurs méthodes pour calculer cette provision, certaines sont courantes, d'autres ne sont pas du tout utilisées. Certaines sont d'usage universel, puisqu'elles sont requises dans l'état annuel. D'autres sont assez complexes, ou extrêmement simples. La plupart de ces méthodes reposent sur quelques principes de base qui ont été combinés de différentes manières, parfois avec une terminologie différente.

Les méthodes déterministes de provisionnement sont utilisées depuis de nombreuses années pour évaluer l'estimation centrale (Best Estimate) des assureurs. Les méthodes les plus utilisées sont les méthodes Chain Ladder et Bornhuetter-Ferguson. Ces méthodes constituent la base de certaines méthodes stochastiques. L'un des avantages des méthodes déterministes est qu'elles produisent une seule réponse (c'est-à-dire le Best Estimate, ou BE) qui est facile à comprendre et à communiquer à l'entreprise. En outre, elles permettent généralement d'inclure un jugement externe dans l'évaluation des provisions de manière assez simple.

La limite des méthodes déterministes est qu'elles ne permettent pas de comprendre le niveau d'incertitude associé au BE. Les tests de scénario et de sensibilité peuvent donner un aperçu de la variabilité des provisions, mais il n'est pas possible d'obtenir une image complète de la volatilité. Pour cela, un modèle stochastique est nécessaire. Le modèle stochastique devrait être utilisé à la fois pour l'estimation du BE et pour l'incertitude associée.

Les méthodes stochastiques fournissent des informations sur la distribution des provisions : au moins les deux premiers moments, c'est-à-dire la moyenne et la variance, mais fréquemment la distribution complète des provisions. Pour ce faire, il faut poser certaines hypothèses stochastiques sous-jacentes au processus observé, en faisant une inférence et en appliquant ensuite une approche analytique (Boostrap) ou de simulation (Monte Carlo). Le principal avantage des méthodes stochastiques est qu'elles fournissent une quantité importante d'informations statistiques, qui n'auraient pas été obtenues avec des méthodes déterministes. Ces informations peuvent être utilisées pour améliorer le processus de prise de décision des résultats possibles. D'autres résultats utiles incluent les intervalles de confiance autour d'une moyenne choisie, les provisions aux extrémités (quantile à 99.5%) et l'analyse de sensibilité.

Cependant, comme le cas des méthodes déterministes, les méthodes stochastiques présentent certains inconvénients. Elles sont exigeantes en données, nécessitant des données historiques pour paramétrer les modèles. Il se peut que l'on ne dispose pas de suffisamment de données ou que l'environnement ait changé, ce qui signifie que l'expérience passée n'est plus représentative de l'expérience future. Il est également important de reconnaître que tous les résultats d'une méthode stochastique sont

sujets à l'erreur de modèle et doivent donc être interprétés avec précaution, car même avec un grand nombre de simulations, l'erreur de modèle peut signifier que certains scénarios négatifs clés peuvent être manqués. En outre, l'utilisation correcte des méthodes stochastiques requiert généralement un degré raisonnable de compétences techniques.

Le développement récent de l'intelligence artificielle, ainsi que l'amélioration incessante de la puissance de calcul des ordinateurs, ont permis de développer des modèles de Machine Learning (ML) (arbre de décision, forêt aléatoire, XGBoost, ...) et de Deep Learning (DL) (Réseaux de neurones, Mixture Density Network, ...) dans le cadre du provisionnement non-vie. L'emploi de ces modèles est un sujet de recherche très actif en ce moment. En effet, ils permettent idéalement de pouvoir concurrencer les méthodes déterministes et stochastiques. En revanche, le manque d'interprétabilité de certains algorithmes de ML et de DL rendant leur utilisation délicate dès lors que les assureurs ont l'obligation de pouvoir expliquer les résultats obtenus.

Le but de ce mémoire est de proposer des modèles de Machine Learning et Deep Learning capables de prendre en compte les tendances et les dépendances temporelles dans un triangle de développement donné en utilisant des données simulées et des données réelles, afin de prédire les provisions et la densité avec précision et de comparer ces modèles avec les méthodes usuelles.

Pour bien aborder ce sujet, nous établirons d'abord les fondamentaux du provisionnement en assurance non-vie, puis nous effectuerons une revue de la littérature sur l'application des méthodes de Machine Learning et de Deep Learning en provisionnement. Nous décrirons ensuite les méthodes usuelles (déterministes et stochastiques) et nous présenterons la théorie des algorithmes utilisés en Machine Learning et en Deep Learning que nous appliquerons sur les données simulées et les données réelles. Par la suite, nous détaillerons les résultats obtenus et nous comparerons notamment les performances de nos modèles de Machine Learning et de Deep Learning aux modèles plus simples et interprétables afin de faire une analyse comparative.

*"**All models are approximations. Essentially, all models are wrong, but some are useful. However, the approximate nature of the model must always be borne in mind**" [*].*
(George Box)

[*] "**Tous les modèles sont des approximations. Essentiellement, tous les modèles sont faux, mais certains sont utiles. Cependant, il faut toujours garder à l'esprit le caractère approximatif du modèle**". La première citation de Box affirmant que "tous les modèles sont faux" se trouve dans un article publié en 1976 dans "*the Journal of the American Statistical Association*", qui contient deux fois cet aphorisme.

Chapitre 1

Provisionnement en assurance non-vie

1.1 Le cadre du provisionnement non-vie

L'assurance est une opération par laquelle l'assureur s'engage à payer une prestation au profit de l'assuré en cas de réalisation d'un événement aléatoire, souvent nommé sinistre en contrepartie d'une prime qui représente une contribution pécuniaire. L'activité d'assurance est marquée par une importante spécificité, à savoir l'inversion du cycle de production.

La figure suivante résume les différentes étapes de la vie d'un sinistre :

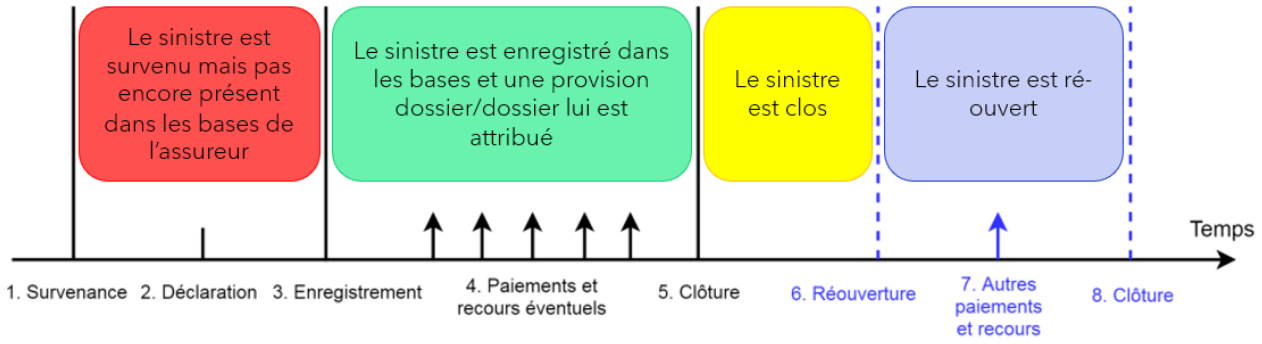

FIGURE 1.1 : Les différentes étapes de la vie d'un sinistre.

On comprend bien que ces potentiels forts écarts de dates vont avoir des impacts sur la gestion de la trésorerie de la compagnie. Il faut donc être en mesure de savoir combien la compagnie va devoir payer, quand et donc choisir ainsi les investissements adéquats à l'actif. C'est le travail des équipes de provisionnement.

Lorsqu'un sinistre survient, il faut le régler. Il faut ainsi provisionner les montants de sinistres futurs pour pouvoir prévoir les règlements. Du coté des entreprises d'assurance, le provisionnement est considéré comme étant une étape cruciale qui permettra d'évaluer les provisions réglementaires liées à la gestion des sinistres, qui représentent une dette de l'assureur envers ses assurés.

1.2 Provision Technique

Une provision est une somme d'argent laissée de côté pour faire face à des événements futurs incertains. Il s'agit d'un stock et évolue en fonction des flux entrants (dotation à la provision) et des flux sortants (reprise de provision) sur un exercice donné.

Il existe différents types de provisions dans une entreprise, nous sommes concentrés dans la suite sur les provisions spécifiques à l'activité opérationnelle d'un assureur Non-Vie : les provisions pour sinistres.

1.2.1 Provision pour sinistre à payer (PSAP)

La principale provision en assurance non-vie est la provision pour sinistre à payer (PSAP). Il s'agit du montant intégral (en principal, le chèque à l'assuré, et en frais) des dépenses nécessaires aux règlements de tous les sinistres survenus et non payés. Il y a des décalages entre la survenance du fait et le règlement effectif du sinistre. La PSAP va permettre de concilier ce décalage et le principe de comptabilisation par exercice de survenance. Elle est calculée exercice par exercice.

1.2.2 Provision dossier/dossier

Une fois un sinistre déclaré, une provision dite dossier/dossier est constituée afin de couvrir les règlements attendus par l'assureur au titre de ce sinistre. L'évaluation peut être faite sur mesure, en fonction des rapports d'expertise. Mais pour les risques de masse, cette provision est généralement forfaitaire et correspond à la charge moyenne constatée pour les dossiers identiques.

La provision dossier/dossier est enregistrée dès que la survenance du sinistre a été portée à la connaissance de l'assureur, son estimation est fondée sur les éléments provisoires à disposition à ce moment-là. La précision de son calcul dépend beaucoup de l'expérience.

1.2.3 Provision IBNR (Incurred But Not Reported)

Un premier type d'IBNR provient de l'aggravation tardive. En effet, l'assureur estime dès la déclaration du sinistre son coût probable. Cependant, il peut y avoir une aggravation intervenant plusieurs mois après la survenance. Pour faire face à ce type de délai, on constitue une provision complémentaire à la provision dossier/dossier car celle-ci ne constitue que la première estimation probable de chacun des sinistres. Cette provision s'appelle IBNeR (Incurred But Not enough Reported).

Le second type d'IBNR provient de la déclaration tardive. En effet, tous les sinistres ne sont pas déclarés instantanément après leur survenance. La déclaration peut être faite après la fin de l'année de survenance. Pour faire face à ce type de délai, l'assureur constitue une autre provision, qui vient compléter la provision dossier/dossier et l'IBNeR, et qui vise à couvrir le coût lié aux sinistres survenus mais pas encore déclarés. C'est l'IBNyR (Incurred But Not yet Reported),

$$\boldsymbol{IBNR = IBNeR + IBNyR}.$$

L'estimation de l'IBNR est différente de celle de la provision dossier/dossier. Elle ne se fait pas au niveau sinistre mais au niveau agrégé par des méthodes statistiques. On a

$$\boldsymbol{PSAP = Provision_{Dossier/Dossier} + IBNeR + IBNyR}.$$

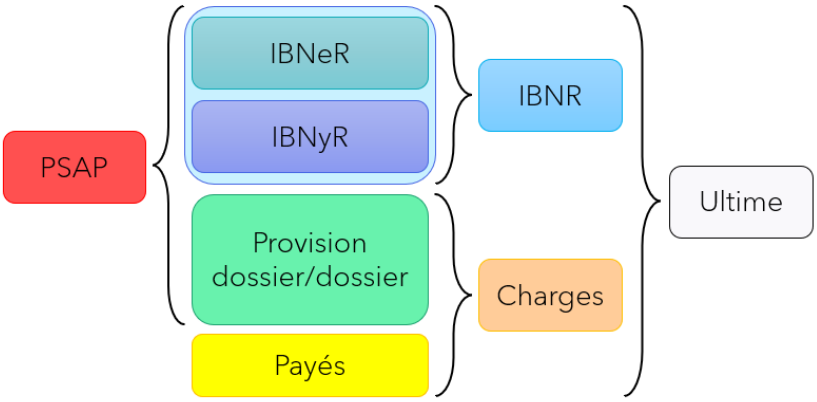

FIGURE 1.2 : Les provisions de sinistres.

1.3 Provisionnement sous l'optique de Solvabilité I & II et IFRS 17

L'évaluation des provisions techniques est réalisée par des différentes approches et normes aux niveaux national (Code des assurances) et international (Solvabilité I, Solvabilité II et IFRS 17).

1.3.1 La réglementation Solvabilité I

Jusqu'au 31/12/2015, la réglementation en vigueur était Solvabilité I, qui exigeait des assureurs qu'ils évaluent au mieux leurs engagements en veillant à constituer des provisions techniques. Un des enjeux primordial de cette réglementation porte sur la méthode de calcul de ces provisions techniques. Des faiblesses ont été aperçus dans cette réglementation, par exemple : tenir en compte d'une vision uniquement rétrospective (la solvabilité future se base sur des chiffres historiques), l'absence de discrimination entre les différents risques, le fait qu'il ne met en considération que le montant espéré des provisions.

La provision pour sinistre à payer en assurance non-vie est la plus importante de toutes les provisions techniques. Elle représente une estimation des dépenses en principal et en frais, tant internes (indemnisation des sinistres de ses clients) qu'externes (règlements effectués au titre de la garantie responsabilité civile) indispensables pour régler les sinistres survenus et non payés. Il s'agit d'une estimation puisque le montant de la charge ultime est sujet à une volatilité donnée due à des événements aléatoires. Une sous-estimation de ces provisions peut avoir des conséquences importantes pour un assureur, allant d'une simple perte à l'insolvabilité. Un niveau de provision très élevé engendre une baisse de résultat pour l'assureur, mais également de la charge fiscale. Ces provisions, évaluées à leur juste valeur, devraient donc expliquer le niveau de risque de l'assureur indépendamment de toute pratique de gestion de résultat.

En solvabilité 1, les sociétés d'assurance doivent également disposer, au-delà de leurs provisions techniques, d'un montant de fonds propres appelé ***Marge de Solvabilité 1*** qui doit être supérieur à un niveau minimal appelé *marge de solvabilité réglementaire* déterminé, soit par rapport au montant annuel des primes ou cotisations, soit par rapport à la charge moyenne annuelle des sinistres. Cette réglementation n'est pas assez harmonisé au niveau européen et on ne constate pas de réelles convergences internationales avec les normes comptables IFRS (*International Financial Reporting Standards*).

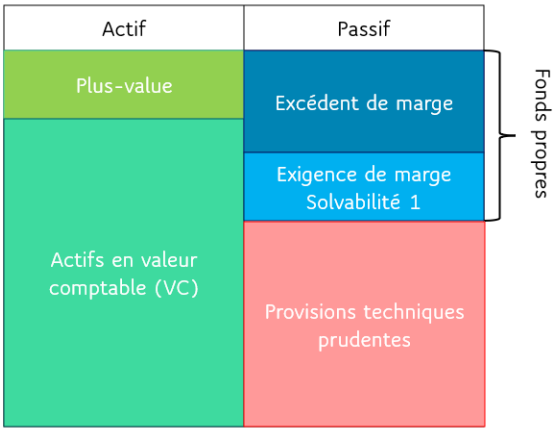

FIGURE 1.3 : Bilan Solvabilité 1.

1.3.2 La réglementation Solvabilité II

1.3.2.1 Solvabilité II

La mise en place effective de Solvabilité II est fixée au 01/01/2016. Elle a pour objectif principal d'adapter au mieux les fonds propres exigés des compagnies d'assurance et de réassurance, afin de confronter les risques auxquels elles sont exposées dans leur activité. L'organisme assureur doit constituer un capital au passif de son bilan afin d'honorer ses engagements vis-à-vis des assurés. Les provisions techniques font partie de ce capital et doivent être rattachées à des classes de risques homogènes, ou Lines of Business (LoB). Elles sont constituées d'une provision Best Estimate et d'une marge pour risque.

En effet, la provision pour sinistre à payer doit être ajustée selon une approche, dite Best Estimate (BE). Elle représente la meilleure estimation des coûts des sinistres c'est-à-dire la valeur actuelle probable des flux de trésorerie futurs. Le calcul de la Best Estimate est fondé sur une modélisation actuarielle qui repose sur des hypothèses réalistes, et des informations crédibles. Le calcul intégre une marge de risque (Risk Margin),

$$\boldsymbol{Provisions\ techniques = BestEstimate + Risk\ Margin.}$$

Le Risk Margin (RM) compense la sous-estimation de la sinistralité et du montant des provisions. Il se calcul par la méthode : *coût du capital*. Cette méthode consiste à calculer le coût de la mobilisation d'un montant de fonds propres éligibles égal aux SCRs nécessaire pour assumer les engagements d'assurance et de réassurance sur toute leur durée de vie. Le niveau des provisions est fixé en référence à une mesure de risque comme la Value-at-Risk ou Tail Value-at-Risk. Dans le cadre de Solvabilité 2, le principe de calcul des provisions techniques se base sur la distinction entre deux catégories de risques :

Les risques couvrables (hedgeable) : une valeur de marché est disponible, le montant des provisions dans ce cas est le prix de marché d'un instrument financier qui répliquerait les flux du contrat d'assurance.

Les risques non couvrables (non hedgeable) : aucune valeur de marché n'est disponible, le montant des provisions techniques dans ce cas correspond au montant qu'une entreprise d'assurance devrait payer actuellement pour transférer ses engagements à un autre assureur.

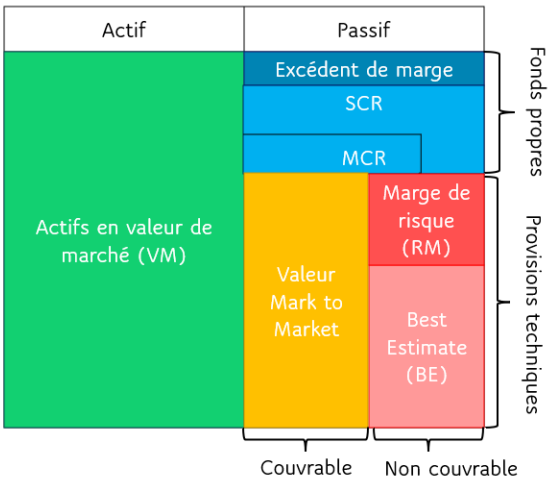

FIGURE 1.4 : Bilan Solvabilité 2.

1.3.2.2 Vision du risque à l'ultime et à 1 an (Ultimate view and one-year view)

Comme nous pouvons le constater, il n'y a pas de définition explicite du risque des provisions dans les spécifications techniques. Pour l'obtenir, nous devons nous référer aux définitions sous Solvabilité II :

Le **SCR** (*Solvency Capital Requirement*) est le niveau de fonds propres à atteindre pour assurer la survie de l'entreprise avec une probabilité de $99,5\%$ à horizon 1 an. Il est calculé indépendamment pour chaque risque, puis agrégé au niveau compagnie en prenant en compte les corrélations entre les risques.

Le **risque de souscription** (*Underwriting risk*) en non-vie est le risque découlant des obligations d'assurance non-vie, en relation avec les périls couverts et les processus utilisés dans la conduite de l'activité.

Le **module de risque de souscription** (*Underwriting risk module*) non-vie prend en compte l'incertitude des résultats des entreprises liée aux obligations d'assurance et de réassurance existantes ainsi qu'aux nouvelles affaires qui devraient être souscrites au cours des 12 mois suivants.

Le **risque de provisions** (*Claims Risk*) : changement de valeur causé par des coûts ultimes pour des obligations contractuelles complètes (sinistres sans coûts administratifs) qui varient de ceux supposés lors de l'estimation de ces obligations. Le risque lié aux provisions ne concerne que les sinistres encourus, c'est-à-dire les sinistres existants (y compris les IBNR et les IBNeR), et provient du fait que la taille des sinistres est plus importante que prévu, des différences dans le délai de paiement des sinistres par rapport aux prévisions, et des différences dans la fréquence des sinistres par rapport aux prévisions.

Globalement, si l'on considère l'ensemble des définitions ci-dessus, on peut résumer que le risque des provisions est le résultat des fluctuations des estimations des sinistres sur une période d'un an. Cette interprétation suscite quelques inquiétudes quant à ce qui doit être considéré comme une "période d'un an".

La **période de choc** (*Shock period*) : la période pendant laquelle un choc est appliqué à un risque.

L'**horizon d'effet** (*Effect horizon*) : la période sur laquelle le choc appliqué à un risque aura un impact sur l'assureur.

En principe, à la fin de la période de choc, le capital doit être suffisant pour que les actifs couvrent la provision technique redéfinie à la fin de la période de choc. La nouvelle détermination des provisions techniques permettrait de prendre en compte l'impact du choc sur les provisions techniques sur l'ensemble de l'horizon temporel des obligations.

Cette définition a conduit aux deux visions du risque des provisions :

Vision à l'ultime : l'incertitude due à tous les paiements futurs P_t.

Vision à un an : l'incertitude du résultat de l'évolution des sinistres, c'est-à-dire l'incertitude des paiements qui auront lieu au cours de l'année P_1 plus la réévaluation des provisions pour sinistres à la fin de l'année.

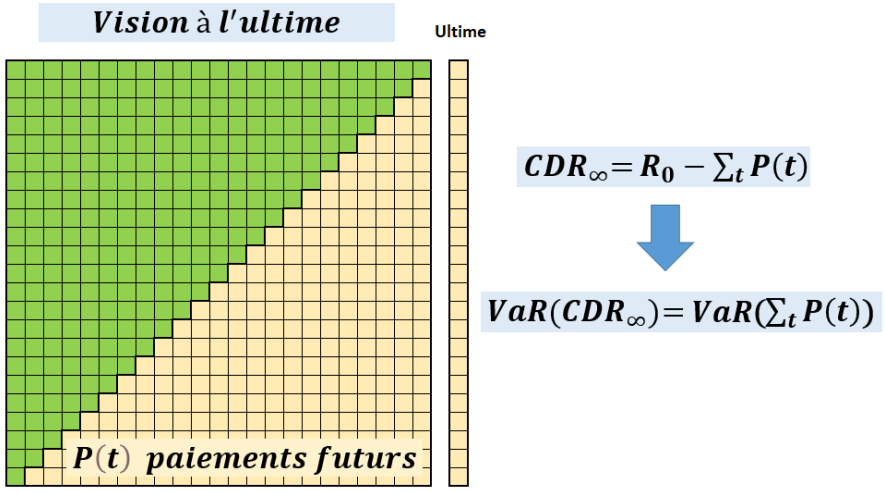

FIGURE 1.5 : Vision à l'ultime.

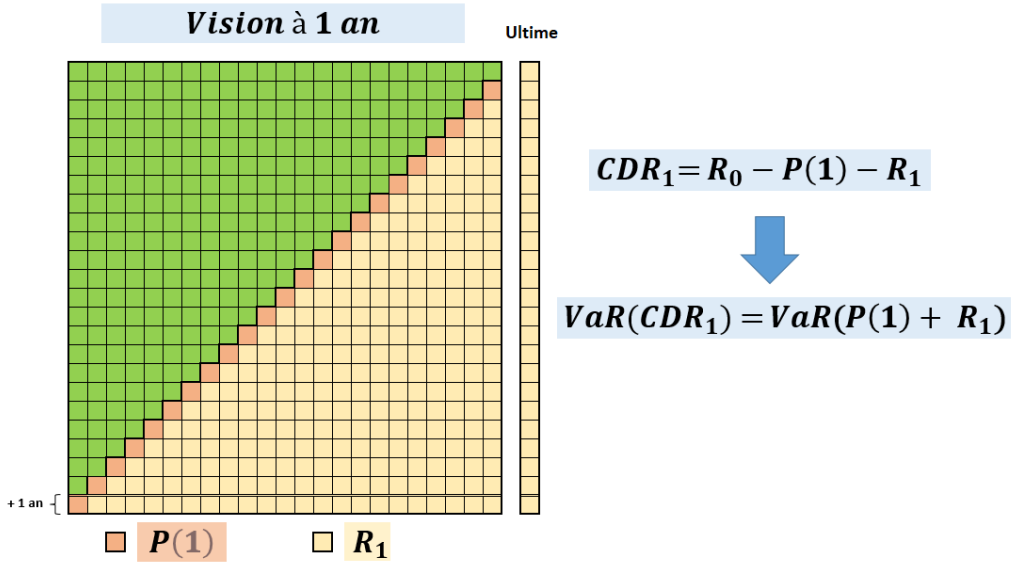

FIGURE 1.6 : Vision à 1 an.

R_0 **:** sont les provisions d'ouverture qui sont connues (c'est-à-dire $VAR(R_0) = 0$).

$P(t)$ **:** sont les paiements durant l'année calendaire t, uniquement pour les sinistres déjà survenus à l'instant d'évaluation $t = 0$.

R_1 **:** sont les provisions de clôture après avoir observé $P(1)$.

CDR_t **:** (*Claim Development Result*) est le résultat de l'évolution des sinistres après t ans à partir de l'instant de l'évaluation, c'est-à-dire la différence entre le réel et l'attendu sur la t-ième période spécifiée.

Idéalement, si les actuaires font vraiment des provisions au mieux en tenant compte de toutes les connaissances possibles, la valeur attendue du CDR pour tout futur t à l'instant de l'évaluation devrait être nulle ("CDR prospectif"). Ceci est opposé au "CDR rétrospectif", qui est le CDR observé après t années et donc non centré sur zéro.

En résumé, la " vision ultime " évalue toutes les trajectoires des provisions possibles (de l'évolution à l'ultime) jusqu'à la liquidation des provisions, tandis que la " vision à 1 an " n'évalue que les différentes trajectoires au cours de la première année et les provisions résultantes qu'un actuaire estimerait après avoir observé chacune de ces trajectoires à un an.

Pour l'estimation du risque lié aux provisions sur un an, on utilise généralement la procédure **Actuary-in-the-Box** .

1.3.3 La norme IFRS 17

La nouvelle norme comptable internationale IFRS 17 a pour but de résoudre les problèmes de comparabilité inhérents à la norme IFRS 4 (une norme qui avait pour objectif de spécifier l'information financière pour les contrats d'assurance émis et les traités de réassurance détenus par la compagnie d'assurance) qu'elle remplacera. Elle a été publiée le 18 mai 2017 et entrera en vigueur au 1^{er} Janvier 2023, impactera le processus de provisionnement, en imposant une segmentation par groupe de risques homogènes. Cette nouvelle norme va imposer à tous les contrats d'assurance d'être comptabilisés de manière identique quelque soit la localisation de la société ayant émis ce contrat. Pour des polices d'une durée de couverture inférieure à 1 ans en assurance non-vie, les provisions techniques se présentent comme la somme d'une estimation des flux de trésorerie futurs et d'un ajustement pour le risque (RA),

$$\boldsymbol{Provisions\ techniques_{(non-vie)}} = \textit{Flux de trésorerie futurs} + \boldsymbol{Ajustement\ pour\ le\ risque}.$$

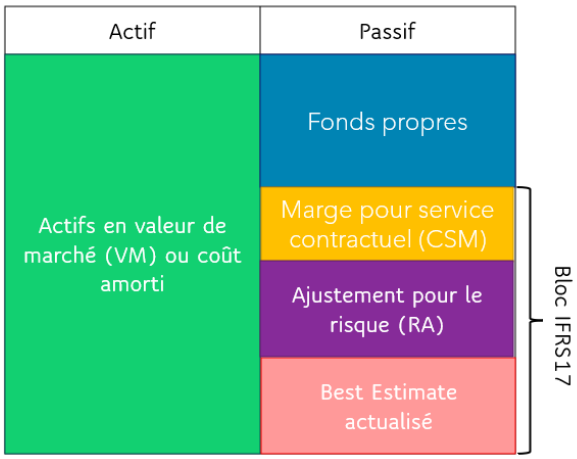

FIGURE 1.7 : Bilan IFRS17.

La norme IFRS 17 a pour objectif de corriger les inadéquations entre l'évaluation du passif et de l'actif présentes sous la norme IFRS 4, en offrant notamment une évaluation économique des provisions techniques, assurant ainsi une meilleure cohérence avec l'évaluation de l'actif.

La norme IFRS17 repose sur des fondamentaux proches de ceux de Solvabilité 2, comme par exemple, l'évaluation du passif d'assurance basée sur une approche prospective des flux de trésorerie en juste valeur. Même si IFRS 17 et Solvabilité 2 possèdent des notions communes, des divergences demeurent : la Risk Margin (RM) sous Solvabilité 2, laisse place au Risk Adjustment(RA) sous IFRS 17, qui correspond au montant que l'entité demande pour prendre à sa charge l'incertitude sur le montant et les dates de versements des flux futurs. Cet ajustement pour risque (RA) permet de prendre en considération le risque non financier. En outre, le passif IFRS 17 introduit *la marge de service contractue*l (CSM) qui permet de lisser le résultat sur toute la durée du contrat et qui correspond aux profits actualisés ainsi aux profits liés aux services futurs fournis par l'assureur.

IFRS 17 possède 3 méthodes de comptabilisation :

- **VFA** (Variable Fee Approach) : applicable aux contrats à participation direct.
- **BBA** (Building Bloc Approach) : la méthode générale.
- **PAA** (Premium Allocation Approach) : la méthode simplifiée.

Pour des polices d'une durée de couverture inférieure à 1 ans en assurance non-vie, la méthode simplifié PAA est utilisée à condition que les évaluations du passif entre les approches BBA et PAA pour ce groupe sont similaires subjectivement (*cf.* paragraphe 53 de la norme IFRS 17).

1.4 Apport de la Data Science en provisionnement non-vie

Les Data Scientists jouent un rôle très important dans le développement des entreprises. Cela dit, la Data Science pousse les compagnies d'assurance à s'y intéresser de plus en plus pour optimiser les processus internes. Depuis le début du XXI° siècle, la recherche autour du Machine Learning et du Deep Learning obtient de grandes avancées. Ses utilisations dans le domaine de la tarification est de plus en plus fréquente afin de compléter ou de concurrencer les estimations données par les Modèles Linéaires Généralisés (GLM).

L'automatisation de la réalisation de diagnostics par l'extraction d'informations ainsi que l'obtention d'un ultime, à l'aide de l'Intelligence Artificielle (AI) façonnée comme une aide à la décision pour les actuaires représente une valeur ajoutée certaine, dont les algorithmes de Machine Learning et de Deep Learning qui font l'objet de plus en plus d'intérêt dans la recherche actuarielle afin de concurrencer les méthodes classiques (Chain Ladder, Mack, ODP, ...). Les méthodes issues de la Data Science ont énormément amélioré les méthodes classiques, que ce soit de façon directe ou indirecte, comme par exemple l'automatisation de la modélisation des segments de sinistres les plus complexes.

L'enquête d'octobre 2020 du Royaume-Uni (MacDonnell (2020)) concernant l'apport de la data science en actuariat a montré qu'il y avait un intérêt quasi universel pour le développement de techniques de Machine Learning dans le domaine des provisions en assurance. Le résultat était plus positif que ce qu'ils ont anticipé. Cela contraste fortement avec l'enquête de 2014 de GIROC (GI Reserving Oversight Committee), qui a révélé qu'il y avait très peu d'importance au Royaume-Uni pour de nouvelles méthodes de provisionnement à l'époque, il semble qu'il y ait eu un grand changement de mentalité ces dernières années. Par conséquent, il a été constaté que les entreprises sont en phase de développement de leurs recherches et ils sont arrivés à un stade très avancé. En revanche les entreprises ont rencontré des difficultés tels que :

- Les contraintes de temps et de ressources, et la difficulté d'acquérir les compétences. Ce n'est pas quelque chose que l'on peut acquérir rapidement avec la charge de travail des actuaires concernant leurs projets et leurs missions.
- La collaboration au sein des entreprises est moins importante. Par exemple, les équipes chargées de la Data Science ou de la tarification ne s'intéressaient souvent pas au provisionnement.

Ce sujet a nécessité un très grand intérêt pour le développement des méthodes de Data Science de la part des équipes de provisionnement plutôt que la direction. Les entreprises internationales ne semblent pas souvent en parler à leurs collègues étrangers et cela est dû au manque de communication entre eux.

Concernant l'enquête faite au Canada (Friedland (2020)) sur le Machine Learning pour le provisionnement en octobre 2020, il y'avait une interrogation de différentes parties prenantes internes et externes. La plupart n'utilisent pas le Machine Learning pour le provisionnement collectifs à cause de la difficulté rencontrée en phase d'apprentissage d'algorithmes. Rarement pour le provisionnement individuel et ils leurs arrivent de trouver des difficultés au cas de l'utilisation aussi. Les modèles appliqués sont souvent GLM, Boost et Taylor McGuire, utilisés seulement dans le contexte de R&D.

Les Actuaires et les Data Scientists diffèrent à plusieurs égards. Le champ d'action d'un data scientist est souvent plus large que celui d'un actuaire au Canada, tandis que les actuaires connaissent mieux les nuances des probabilités et statistiques que les data scientists avec des acquis très limités en Machine Learning. Parmi les obstacles rencontrés dans l'utilisation du Machine Learning pour le provisionnement en Canada :

- Difficile de mesurer l'incertitude de prédiction dans le domaine de provisionnement IARD par les méthodes de Machine Learning et Deep Learning. En outre, il y a toujours d'autres demandes qui ont la priorité comme par exemple la réglementation IFRS 17.
- Le défi de trouver l'utilisation la plus appropriée du Machine Learning et du Deep Learning d'une façon dont elles s'intègrent le mieux dans le cadre de la gouvernance des provisions, par exemple : segmentation et test de scénario.

Malgré les méthodes innovantes de la data science, l'acceptation totale du Machine Learning et du Deep Learning affronte encore deux obstacles principaux. Le premier d'entre eux est paradoxal puisqu'il est lié au Big Data. En effet, si un grand nombre de données améliore sensiblement les résultats des algorithmes du Machine Learning et Deep Learning, il demande également une puissance de calcul toujours plus élevée, alors cela nécessite un temps de calcul important contrairement au méthodes classiques. Le second obstacle est lié au manque d'interprétabilité de certains algorithmes de Machine Learning et Deep Learning, comme XGBoost et les Réseaux de Neurones. Il est parfois difficile de justifier les choix d'algorithmes agissant avec un effet *Boîte Noire* pour lesquels l'inférence des variables dans le résultat est difficilement quantifiable.

1.5 Contexte du mémoire

La modélisation et l'estimation des provisions techniques en assurance non vie sont des problématiques centrales de l'actuariat, les assureurs sont obligés de détenir des provisions pour faire face aux sinistres en cours lorsqu'ils arrivent à échéance. Ces sinistres représentent souvent une part importante du passif de l'assureur. Par conséquent, le fait de ne pas détenir un capital économique suffisant pour faire face à ces créances peut entraîner des problèmes de liquidité. L'importance de ce type de risque a engendré des réglementations afin d'avoir des provisions suffisantes pour faire face aux engagements de sinistres.

De ce fait, il est non seulement important de trouver une meilleure estimation centrale, mais aussi la modélisation de la distribution qui aidera à évaluer la volatilité de ces engagements et permettra aux assureurs d'allouer efficacement un capital pour satisfaire les exigences réglementaires. En outre, l'estimation des sinistres et leur distribution est importante pour la tarification, car les primes doivent être fixées de manière à couvrir les sinistres en fonction d'une marge bénéficiaire.

Plusieurs modèles statistiques ont été utilisés aujourd'hui et dans le passé pour modéliser les sinistres en cours. La plupart des modèles, même aujourd'hui, se sont concentrés sur les Triangles de developpement agrégées, tels que la méthode Chain Ladder, introduite dans les années 1960, utilisaient un algorithme déterministe pour fournir une estimation centrale. Avec le manque de stochasticité fourni par le modèle Chain Ladder, le modèle Mack a été introduit en 1993 pour aider à quantifier l'incertitude des sinistres en cours. D'autres méthodes détérministes ont été utilisés à cette époque, notamment les modèles Bornhuetter-Ferguson et Cape Cod.

Avec l'amélioration de la puissance de calcul des ordinateurs, les modèles linéaires généralisés (GLM) sont devenus courants dans la littérature actuarielle, à partir des années 1990. Le modèle GLM Poissonnien de (Renshaw et Verrall (1998)), a permis d'obtenir des résultats plus précis, d'avoir la distribution des sinistres, ce qui a rendu les prévisions probabilistes. Les algorithmes de Machine Learning font l'objet de plus en plus d'intérêt dans la recherche actuarielle afin de concurrencer les méthodes classiques, En 2017, le groupe ASTIN propose d'utiliser un MLP (Multi-Layer Perceptron), la forme classique des réseaux de neurones en utilisant à la fois les paiements incrémentaux et la PSAP parmi les variables d'entrées. En 2018, Mario Wuthrich (2018) a utilisé les arbres de décisions pour l'estimation du nombre de paiements par sinistre.

Par ailleurs, Kevin Kuo propose en 2019 un modèle de provisionnement s'appuyant sur un type de réseau de neurones particulier, Deep Triangle (Kuo (2019)). Ce modèle se distingue notamment des autres par l'emploi de structures récurrentes. Kevin Kuo en illustre les performances par une analyse comparative avec d'autres méthodes stochastiques et déterministes. En effet, Kevin Kuo a d'utilisé des réseaux de neurones récurrents afin d'estimer simultanément les paiements incrémentaux et la PSAP. Ce type de réseaux neuronaux d'une complexité supérieure aux MLP permet au modèle d'apprendre des tendances sur des séquences temporelles. En 2020, Gabrielli (2020a) a utilisé un réseau neuronal résiduel (Residual Neural Network) pour renforcer un modèle GLM. Bien qu'ils soient prometteurs, plusieurs lacunes sont apparues dans les réseaux neuronaux et leur mise en œuvre actuelle au niveau de la prédiction des sinistres. Tout d'abord, l'accent est peu mis sur la prévision de la distribution des sinistres, et se concentre plutôt sur les estimations centrales. En plus, les modèles de réseaux neuronaux sont considérés comme une boîte noire en raison de leur complexité, ils sont donc beaucoup moins faciles à interpréter que les modèles GLM.

En raison de leur manque d'interprétabilité, les réseaux de neurones sont confrontés à un risque plus élevé (sur-apprentissage et problème de la disparition du gradient), ce qui a ralenti leur acceptation dans l'actuariat. Pour renforcer leur acceptation, en 2019 Wuthrich (2019) a développé le reseaux de neurones actuariels combinés (CANN) qui a intégré un GLM dans l'architecture du réseau neuronal pour une amélioration de l'interprétabilité et de la stabilité des réseaux neuronaux.

L'application des réseaux neuronaux au provisionnement non-vie est très récente, Mulquiney (2006) étant le premier à les appliquer. Leur application s'est concentrée sur l'amélioration des modèles GLM existants (Gabrielli (2020a)). Par contre, Kuo (2020) s'est attaché à fournir une prévision probabiliste du Best Estimate non basée sur un GLM. Les réseaux de neurones ont été appliqués à la fois à des triangles de développement et à des sinistres individuels, la plupart d'entre eux démontrant la supériorité de leurs modèles par rapport aux modèles usuelles pour le calcul du Best Estimate.

En 2020, Kuo (2020) a mis en place l'application d'un réseau de neurones de type MDN (Mixture

Density Networks) dans le cadre de prévision des sinistres individuels. Le modèle de mélange gaussien utilisé dans le MDN, avec suffisamment de composants, peut prédire n'importe quelle distribution avec une précision souhaitée. Un autre avantage des MDN est que l'estimation centrale peut être déduite directement des paramètres du mélange gaussien prédits. En 2021, M. Taher Al-Mudafer (2021) a développé deux modèles (*MDN* et *ResMDN*) pour la prédictions des triangles de developpement agrégés, qui s'interesse non seulement d'estimer l'estimation centrale, mais aussi à determiner la distribution probabiliste des sinistres. Par suite, ces modèles ont été comparés avec le modèles ccODP (Cross-Classifed Over-Dispersed Poisson) afin de comparer leur performances.

Les méthodologies appliquées dans ce mémoire tire principalement son inspiration des deux derniers articles : *Stochastic loss reserving with mixture density neural networks* de M. Taher Al-Mudafer (2021) et *Individual Claims Forecasting with Bayesian Mixture Density Networks* de Kuo (2020), tout en adaptant le problème et les techniques utilisées aux données que nous avons à notre disposition. Ainsi, on comparera les résultats du modèle MDN de Deep Learning, d'une part avec les modèles de Machine Learning, à savoir Decision tree, XGBoost et Random Forest, et d'autre part avec les méthodes classiques de provisionnement (Chain Ladder, Mack et GLM ODP).

Au sein de ce mémoire, l'objectif est de mettre en place des méthodologies innovantes de Machine Learning et de Deep Learning afin de concurrencer les méthodes usuelles liées au Chain-Ladder et au GLM Poisson surdispersé (ODP) pour l'estimation du Best Estimate et la densité des sinistres. Nous calibrerons également le modèle XGBoost ainsi que l'algorithme des Random Forest afin de mesurer la valeur ajoutée d'un modèle plus complexe. Les données d'entrée sont des triangles de données agrégées, considérées comme étant un ensemble de séries temporelles. La masse importante de données simulées avec Synthetic Claims Simulator (R)[*†] dans des environnements différentes (une branche à développement court, une branche à développement court, une branche avec une variation de la cadences des sinistres, une branche à développement long avec un choc d'inflation et une branche à développement long avec une forte volatilité) permettent en effet de tirer parti des atouts des méthodes de Machine Learning et de Deep Learning. Cependant, l'ajustement des modèles est effectué en utilisant une approche de type Rolling-Origin Cross-Validation à l'aide d'une validation croisée 5-Fold.

Les langages de programmation utilisés dans ce mémoire sont Python et R. Pour répondre au défi de temps et de la puissance de calcul pendant la phase d'apprentissage et de calibration des modèles de Machine Learning et de Deep Learning, nous utiliserons un RDP (Remote Desktop Protocol), c'est une machine virtuelle d'Addactis France très puissante possédant les caractèristiques suivantes :

- Système d'exploitation : Windows Server 2019 Standard 64 bits.
- Processeur : Intel(R) Xeon(R) 6226R CPU 2.90 Ghz (4 coeurs pour 8 threads).
- Mémoire vive (RAM) : 64 GB.

[*]https ://cran.r-project.org/web/packages/SynthETIC/vignettes/SynthETIC-demo.html

[†]SynthETIC peut être utilisé pour générer un historique de sinistres d'assurance générale avec des hypothèses de distribution réalistes et cohérentes avec l'expérience d'un portefeuille spécifique (mais anonyme) de responsabilité civile automobile.

1.6 Méthodes usuelles de provisionnement

On distingue deux types de méthodes de provisionnement en assurance non-vie, les méthodes individuelles et les méthodes agrégées.

Les méthodes individuelles se différencient les unes des autres selon la façon dont les données sont représentées, certains algorithmes utilisent seulement les données quantitatives du sinistre comme les paiements ou les charges, tandis que d'autres utilisent également les données qualitatives du sinistre.

Les méthodes agrégées présentent les données sous forme des triangles de développement pour projeter les paiements ou/et les charges futurs liés aux sinistres, qui peuvent être construits de manière cumulée ou incrémentale, dans lequel les charges de sinistres sont regroupées par période (année, trimestre, ...) de survenance et par période (année, trimestre, ...) de développement.

1.6.1 Notion de triangles de développement (run-off triangles)

Il existe différents types de triangles de développement ou de liquidation, par exemple : les triangles de règlements ou de charge, les triangles de provision dossier/dossier et les triangles de nombre de sinistres. dans la suite de notre mémoire, on va se limiter sur les méthodes agrégées. Les sinistres sont représentés de la façon suivante :

- les lignes correspondent aux périodes de survenance des sinistres : $i \in [\![1, I]\!]$.
- Les colonnes correspondent aux périodes de développement : $j \in [\![1, J]\!]$.
- $X_{i,j}$ le montant incrémental d'intérêt.
- $C_{i,j}$ le montant cumulé d'intérêt.
- Les diagonales correspondent à des périodes calendaires $i + j$.
- R_i le résultat ultime des provisions à constituer pour les sinistres survenus durant la période i.

On a $C_{i,j} = \sum_{t=1}^{j} X_{i,t}$ et $X_{i,j} = C_{i,j} - C_{i,j-1}$. l'augmentation croissante d'information au cours du temps, à une période calendaire $i + j$ donnée, permet de connaître que le triangle supérieur $D_t = \{X_{i,j}, i + j \leq t, 1 \leq j \leq J\}$ qui représente les informations connues à la date de l'inventaire.

Le triangle inférieur représente les informations que nous cherchons à estimer et à prédire.

L'objectif est alors d'estimer les valeurs du triangle inférieur afin de constituer la PSAP par période de survenance.

		Année de développement						
		1	**2**	**...**	**j**	**...**	**J**	**Résultat ultime**
Année de survenance	**1**	$C_{1,1}$	$C_{1,2}$	...	$C_{1,j}$	...	$C_{1,J}$	**0**
	2	$C_{2,1}$	$C_{2,2}$	...	$C_{2,j}$	...	$\hat{C}_{2,J}$	$R_2 = \hat{C}_{2,J} - C_{2,J-1}$
	...	...	...	...	...	...	...	...
	i	$C_{i,1}$	$C_{i,2}$	...	$\hat{C}_{i,j}$	...	$\hat{C}_{i,J}$	$R_i = \hat{C}_{i,J} - C_{i,j-1}$
	...	...	...	...	...	...	...	...
	I	$C_{I,1}$	$\hat{C}_{I,2}$	...	$\hat{C}_{I,j}$	...	$\hat{C}_{I,J}$	$R_I = \hat{C}_{I,J} - C_{I,1}$

Triangle supérieur

Triangle inférieur

$$PSAP = \sum_{i=1}^{I} R_i$$

FIGURE 1.8 : Exemple de triangle de développement des paiements cumulés.

1.6.2 Méthodes déterministes

1.6.2.1 Méthode de Chain Ladder

▷ Principe

La méthode Chain Ladder est une méthode déterministe qui est la plus utilisée pour le calcul des provisions. Elle s'applique à des triangles de paiements cumulés ou des triangles de charges afin d'estimer le triangle inférieur qui représente les éléments futurs. Elle repose sur deux hypothèses :

- ***Hypothèse 1*** : Les années de survenance sont indépendantes.
- ***Hypothèse 2*** : Si les ratios $\frac{C_{i,j+1}}{C_{i,j}}$ ne dépendent pas de l'année de survenance i, pour tout $i = 1, ..., I$ et $j = 1, ..., J-1$, alors il existe des paramètres $\lambda_1, \cdots, \lambda_{J-1}$ tels que $C_{i,j+1} = \lambda_j C_{i,j}$.

Les facteurs $\lambda_1, \cdots, \lambda_{J-1}$ sont appelés facteurs de développement, coefficients de passage ou encore cadences (link ratios). On calcule alors le facteur de développement lié à l'année de développement j par

$$\lambda_j = \frac{\sum_{i=1}^{I-j} C_{i,j+1}}{\sum_{i=1}^{I-j} C_{i,j}}, \forall j \in [\![1, J]\!].$$

Une fois les facteurs de développement estimés, on calcule les paiements cumulés par la formule suivante

$$\hat{C}_{i,j} = \prod_{k=j}^{J-i} \lambda_k \times C_{i,I-i}.$$

On complète après le triangle grâce aux $\hat{C}_{i,j}$ estimés. On obtient ainsi les provisions à l'ultime pour l'année de survenance i

$$R_i = \hat{C}_{i,J} - C_{i,j-1}.$$

Ainsi les provisions à l'ultime pour l'ensemble des sinistres est donnée par

$$R_{totale} = \sum_{i=1}^{I} R_i.$$

La méthode de Chain Ladder ne fait aucune hypothèse sur la loi de probabilité suivie par les coûts et les fréquences des sinistres, et donc elle ne permet pas d'évaluer la précision de l'estimation obtenue.

▷ **Validation du modèle**

L'hypothèse d'indépendance des ratios $\frac{C_{i,j+1}}{C_{i,j}}$ par rapport à l'année de survenance i peut être vérifiée à l'aide de graphiques **C-C plots** : si $\forall j \in [\![1, J-1]\!]$ fixé, il existe λ_j tel que $C_{i,j+1} = C_{i,j} \cdot \lambda_j$ alors les couples $(C_{i,j}, C_{i,j+1})_{i=1\ldots n-j}$ doivent être approximativement alignés sur une droite passant par l'origine.

Il existe une autre méthode pour valider le modèle appelé **d-triangle**, qui se fait à l'aide du triangle formé des facteurs individuels $\lambda_{i,j} = \frac{C_{i,j+1}}{C_{i,j}}$. Les facteurs de chaque colonne j $\in [\![1, J-1]\!]$, du d-triangle doivent être approximativement constants, donc l'hypothèse 2 de Chain Ladder est vérifiée si le coefficient de variation est suffisamment faible. Les valeurs atypiques peuvent-être exclus du calcul des facteurs de développements λ_j.

1.6.2.2 Méthode de Bornhuetter-Ferguson

▷ **Principe**

La méthode Bornhuetter-Ferguson est une méthode déterministe, développée en 1972 par Bornhuetter et Ferguson (1972). Sa particularité est d'estimer l'ultime et les provisions en y intégrant une information exogène au triangle de developpement cumulé : les provisions se déterminent à partir d'une estimation des ultimes appelés ultimes a priori qu'on notera α, en appliquant un taux de liquidation croissant sur les années de développement. Elle assure donc une meilleure stabilité des estimations et est donc plus adaptée aux triangles dont les incréments sont instables. On suppose dans cette partie que $I = J = N$, La méthode Bornhuetter-Ferguson repose sur l'hypothèse suivante :

Hypothèse : Modèle multiplicatif verifiant

$$\forall i, j \in \{1, \cdots, N\}, C_{i,j} = \alpha_i \gamma_j,$$

où $\alpha_1, \cdots, \alpha_N$ représentent les charges à l'ultime prévisibles pour chaque année de survenance i, et $\gamma_1, \cdots, \gamma_N$ les cadences de paiements cumulés.

Cette méthode requiert un avis d'expert dans le but de déterminer le loss ratio donné par $\frac{S}{P}$ correspondant au quotient entre le coût des sinistres et les primes reçues par l'assureur. L'ultime a priori pour chaque année de survenance i est calculé par la formule suivante

$$\alpha_i = \frac{S}{P} \times P_i,$$

avec P_i la prime pure reçue l'année i. Les cadences de paiements cumulés γ_i pour chaque année de survenance i sont calculées par la formule suivante

$$\gamma_k = \prod_{i=k}^{N-1} \frac{1}{\lambda_k} \quad \text{et} \quad \gamma_N = 1,$$

avec λ_i sont les facteurs de développement calculés par la méthode Chain Ladder. Enfin, les états futurs sont estimés par

$$\hat{C}_{i,j} = C_{i,N-i} + (\gamma_j - \gamma_{N-i})\, \alpha_i.$$

Les provisions de cette méthode seront obtenues par la différence entre le coût total estimé et la dernière valeur observée

$$R_i = \hat{C}_{i,N} - C_{i,N-i} = \alpha_i (1 - \gamma_{N-i}).$$

Contrairement aux autres méthodes déterministes, cette méthode permet d'ajouter de l'information à travers de jugements d'experts pour la détermination des ultimes a priori (charges à l'ultime exogènes).

La détermination du Loss ratio est fortement subjective parce que cela dépend beaucoup de l'expert. En revanche, si ce Loss ratio n'est pas bien estimé, nous courrons alors le risque d'estimer une charge ultime des sinistres fortement biaisée puisqu'elle dépend fortement de ce ratio. Donc la méthode ne garantit pas toujours une stabilité du résultat année après année.

1.6.3 Méthodes stochastiques

1.6.3.1 Méthode de Mack

▷ **Principe**

La méthode de Mack (1993) est une méthode stochastique, qui est considérée comme la première méthode qui fait intervenir la notion d'incertitude dans la méthode déterministe de Chain Ladder. C'est-à-dire elle mesure l'incertitude associée à la prédiction du montant des provisions que doit faire l'assureur. Cette approche mesure la volatilité des provisions déterministes de Chain Ladder. Elle repose sur les hypothèses suivantes :

- ***Hypothèse 1*** : Les paiements cumulés sont indépendants lorsque les années de survenance sont différentes
$$\forall i \neq k, C_{i,j} \perp C_{k,j}.$$
- ***Hypothèse 2*** : les lignes de règlements cumulés $(C_{i,j})_j$ comme des processus markoviens et il existe des facteurs de développement λ indépendants de l'année de survenance verifiant
$$\mathbb{E}\left[C_{i,j+1} \mid C_{i,1}, \ldots, C_{i,j}\right] = \mathbb{E}\left[C_{i,j+1} \mid C_{i,j}\right] = C_{i,j} \cdot \lambda_{i,j}.$$
Ainsi, nous obtenons avec la méthode de Mack le même estimateur des facteurs de développement que celui du Chain-Ladder.
- ***Hypothèse 3*** : Il existe des paramètres $\sigma_{i,j}$ tels que
$$\mathrm{Var}\left[C_{i,j+1} \mid C_{i,1}, \ldots, C_{i,j}\right] = \mathrm{Var}\left[C_{i,j+1} \mid C_{i,j}\right] = C_{i,j} \cdot \sigma_{i,j}^2.$$

En supposant que $I = J = N$, l'estimation de $\sigma_{i,j}$ est donnée par la formule suivante

$$\hat{\sigma}_j^2 = \frac{1}{N-j-1} \times \sum_{i=1}^{N-j} C_{i,j} \times \left(\frac{C_{i,j+1}}{C_{i,j}} - \hat{\lambda}_j\right)^2, \forall j \in [\![1, N-2]\!]$$

et

$$\hat{\sigma}_{N-1}^2 = \min\left(\frac{\hat{\sigma}_{N-2}^4}{\hat{\sigma}_{N-3}^2}, \min\left(\hat{\sigma}_{N-3}^2, \hat{\sigma}_{N-2}^2\right)\right).$$

Pour calculer une estimation des paiements cumulés à l'ultime on utilise la formule suivante

$$\hat{C}_{i,N} = \left(\prod_{k=N-i}^{N-1} \hat{\lambda}_k\right) C_{i,N-i}.$$

▷ **Estimation des erreurs**

L'incertitude sur la prédiction des provisions totales R peut être quantifiée par : la moyenne quadratique des erreurs de prédiction (MSEP) pour chacune des provisions estimées par année de survenance

$$\text{MSEP}(\hat{R}) = \mathbb{E}\left[(\hat{R} - R)^2 \mid C_{i,j}, i + j \leq N + 1\right].$$

On peut obtenir une estimation explicite de $\text{MSEP}(\hat{R})$ sous les trois hypothèses de Mack. Cette valeur est aussi appelée variance totale :

$$\widehat{\text{MSEP}}(\hat{R}) = \sum_{i=2}^{N} \widehat{\text{MSE}\,P}\left(\hat{R}_i\right) + \hat{C}_{i,N}\left(\sum_{l=i+1}^{N} \hat{C}_{l,N} \sum_{j=N-i+1}^{N-1} \frac{2\hat{\sigma}_j^2}{\hat{\lambda}_j^2 \sum_{k=1}^{N-j} C_{k,j}}\right)$$

avec

$$\widehat{\text{MSE}P}\left(\hat{R}_i\right) = \hat{C}_{i,N}^2 \sum_{j=N-i-1}^{N-1} \frac{\hat{\sigma}_j^2}{\hat{\lambda}_j^2}\left(\frac{1}{\hat{C}_{i,j}} + \frac{1}{\sum_{k=1}^{N-j} C_{k,j}}\right).$$

Afin de construire un intervalle de confiance autour des provisions estimées il faut supposer une loi de distribution des provisions. La loi la plus couramment utilisée est la loi Normale de moyenne R_i et d'écart-type $\sqrt{MSEP\left(R_i\right)}$, bien que la loi log-normale soit parfois plus appropriée. En injectant les estimateurs des provisions et de l'erreur quadratique moyenne, nous obtenons un intervalle de confiance de risque α

$$IC_{1-\alpha} = \left[\hat{R}_i - q_{1-\frac{\alpha}{2}} \times \sqrt{\widehat{MSEP}\left(\hat{R}_i\right)}, \hat{R}_i + q_{1-\frac{\alpha}{2}} \times \sqrt{\widehat{MSEP}\left(\hat{R}_i\right)}\right],$$

où $q_{1-\frac{\alpha}{2}}$ est le quantile d'ordre $1-\frac{\alpha}{2}$ de la loi Normale $\mathcal{N}(0,1)$. Cet intervalle de confiance est également valable pour les provisions totales.

▷ **Validation du modèle**

Les deux hypothèses relatives au Chain Ladder peuvent être validés à l'aide des graphiques C-C plots et par le principe du d-triangle. Pour l'hypothèse ajoutée par la méthode Mack, sa vérification consiste à afficher graphiquement les résidus du modèle pour j fixé, les résidus étant définis par

$$\varepsilon_{i,j} = \frac{C_{i,j+1} - \lambda_j C_{i,j}}{\sigma_j \sqrt{C_{i,j}}},$$

les résidus ne doit laisser apparaître aucune structure particulière.

1.6.3.2 Modele de GLM Poisson surdispersé (GLM ODP)

Le modèle GLM Poisson surdispersé a été introduit en 1998, par Renshaw et Verrall (1998) dans le contexte du provisionnement. Sa particularité est de donner les mêmes valeurs pour les provisions futures que le modèle déterministe de Chain Ladder. Bien que la méthode de Mack propose une façon de quantifier l'erreur de prédiction, elle ne fournit pas une distribution des provisions qui est le but ultime d'une méthode de provision surtout dans la réglementation de Solvabilité 2. Les modèles linéaires généralisés permettent de faire cela, en supposant explicitement la distribution paramétrique des règlements.

▷ **Présentation des Modèles linéaires généralisés (GLM)**

On dispose de n observations indépendantes $(y_i, x_i)_{i=1,...,n}$ òu $x_i \in \mathbb{R}^r$ et $y_i \in \mathbb{Y} \subset \mathbb{R}$ pour tout $i \in \{1, ..., n\}$. On définit la matrice *design* X comme suit

$$X = \begin{pmatrix} x_1^1 & \cdots & x_1^r \\ \vdots & & \vdots \\ x_n^1 & \cdots & x_n^r \end{pmatrix} = (X^1, ..., X^r) \text{ et } Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix},$$

ou $X^k, k \in \{1, ..., r\}$ sont les variables explicatives. Le modèle linéaire généralisé est une extension du modéle linéaire permettant de traiter des observations dont la loi de probabilité appartient à une famille de lois élargie. Soient $y_1, y_2, ..., y_n$ $n-$observations indépendantes d'une variable quantitative. D'autre part, pour chaque observation i, on dispose de p variables explicatives $(x_i^1, x_i^2, ..., x_i^p)$ réelles. On cherche à "expliquer" y_i comme une fonction des $(x_i^1, x_i^2, ..., x_i^p)$. Le modèle linéaire généralisé est la donnée d'une loi de probabilité pour les y_i et d'une fonction g appelée fonction de lien qui permet d'établir une relation non linéaire entre l'espérance de la variable à expliquer et les covariables

$$g(E[y_i|x_i]) = x_i^T \beta.$$

• **Famille exponentielle naturelle**

La famille exponentielle naturelle est une famille de lois de probabilité qui contient entre autres des lois aussi usuelles que la loi normale, Poisson, ... Ces lois ont en commun une écriture sous forme exponentielle qui va permettre d'unier la présentation des résultats. Soit f_Y la densité de probabilité de la variable Y. f_Y appartient à la famille exponentielle naturelle si elle s'écrit sous la forme

$$f_Y(y) = exp(\tfrac{1}{a(\phi)}(y\theta - b(\theta)) + c(y, \phi)),$$

c est une fonction dérivable, b est trois fois dérivable et sa dérivée première b' est inversible. Le paramètre θ réel est appelé paramètre naturel de la loi. ϕ est un paramétre appelé paramétre de nuisance ou de *dispersion*. Dans ce cas on a

$$E[Y] = \mu = b'(\theta) \text{ et } V(Y) = b''(\theta)a(\phi).$$

• **Choix de la fonction de lien**

Toute bijection de l'espace de $E[Y]$ dans $\mathbb{R}$ peut être choisie comme fonction de lien. Cependant, très souvent on choisit comme fonction de lien la fonction qui transforme l'espérance $E[Y]$ en paramétre naturel : $g = (b')^{-1}$, g ainsi définie est appelée fonction de lien canonique. Cela permet d'assurer la convergence de l'algorithme d'estimation utilisé de Newton-Raphson vers le maximum de vraisemblance. Le Tableau suivant présente quelques modèles linéaires usuels. A chaque choix de la loi de $Y|X = x$ correspond une fonction de lien canonique g(.) qui donne son nom à la regression.

Choix de la loi de $Y\|x$	Bernoulli/Binomial	Poisson	Gamma	Gaussienne
Fonction de lien canonique	$g(\mu) = log(\frac{\mu}{1-\mu}) = logit(\mu)$	$g(\mu) = log(\mu)$	$g(\mu) = \frac{-1}{\mu}$	$g(\mu) = \mu$
Nom du lien	*logit*	*log*	réciproque	identité

TABLE 1.1 : GLM usuels. $g(E[Y|x]) = x^T\beta$.

• **La deviance**

Pour avoir une idée de la qualité du modèle, on compare la vraisemblance d'un autre modèle $l_{[m]}$ à celle d'un modèle de référence (le modèle "*saturé*") $l_{[sat]}$, ce modèle posséde autant de paramètres que d'observations et estime donc y par $\widehat{y}$, dans ce cadre : $\widehat{E[y_i|x_i]} = y_i$.
La deviance d'un modèle $[m]$ par rapport au modèle saturé $[sat]$

$$D_{[m]} = 2(l_{[sat]} - l_{[m]}) \geq 0 \ .$$

Si le modèle $[m]$ a une $l_{[m]}$ proche de $l_{[sat]}$, alors on préfere le modèle $[m]$: on rend $D_{[m]}$ "petit" pour selectionner le modèle $[m]$. La déviance est une quantité positive d'autant plus petite que le modèle est riche et s'ajuste bien. La déviance vaut 0 pour le modèle le plus riche, i.e. le modèle saturé.

• Qualité d'ajustement

On mesure la qualité d'ajustement du modèle grâce au coéffient de détermination ajusté R_a^2 construit par analogie avec le coéffient de détermination classique R^2. Pour cela, on compare la déviance du modèle nul (à un seul paramètre : intercept) $D_{[m_0]}$ avec celle du modèle qui nous intéresse $[m]$

$$R_a^2 = \frac{D_{[m_0]} - D_{[m]}}{D_{[m_0]}}.$$

Plus R_a^2 est proche de 1, meilleur est l'ajustement du modèle : le modèle s'ajuste bien aux données.

• Choix de modèle : Critère AIC et BIC

Un modèle sera qualifié de bon si sa déviance est proche de celle du modèle saturé et qu'il est construit avec un faible nombre de paramètres. Des critères pénalisés permettent de prendre en compte ces deux contraintes antagonistes. On choisira le modèle qui minimise ces critères.
Le plus célèbre d'entre eux est le critère **AIC** (Akaike Information Criterion)

$$AIC(M) = D(M) + 2r_1 + C_{te},$$

avec r_1 est le rang de la matrice de covariable X. L'AIC est d'autant plus faible que la log-vraisemblance est élevée et que le nombre de paramètres est petit et permet donc d'établir un ordre sur les modèles en prenant en compte les deux contraintes.
Le critère **BIC** (Bayesian Information Criterion) qui pénalise plus le sur-ajustement est défini par

$$BIC(M) = D(M) + \log(n)r_1 + C_{te}.$$

• Validation du modèle

Le test d'adéquation de la déviance permet de comparer la vraisemblance du modèle à celle du modèle saturé. En effet, plus la déviance est faible, meilleur est le modèle en termes d'ajustement.

L'analyse des résidus permet d'affiner le modèle dans une certaine mesure. Elle permet de détecter des effets non linéaires ou encore de détecter des individus atypiques ou aberrants. On distingue plusieurs types de résidus :

- les résidus ≪bruts ≫ $r_{ij} = X_{ij} - \hat{X}_{ij}$.
- les résidus non scalés de Pearson $r_{ij}^{(p)} = \frac{y_{ij} - \hat{\mu}_{ij}}{\sqrt{V(\hat{\mu}_{ij})}}$.
- les résidus scalés de Pearson $r_{ij}^{(p')} = \frac{y_{ij} - \hat{\mu}_{ij}}{\sqrt{\hat{\phi} \cdot V(\hat{\mu}_{ij})}}$.
- les résidus de Pearson ajustés pour tenir compte du nombre de degrés de liberté

$$r_{ij}^{(p'')} = \sqrt{\frac{n}{n-p}} \cdot r_{ij}^{(p')},$$

où n est le nombre d'observations et p le nombre de degrés de liberté du modèle.

Les graphiques de résidus permettent de valider partiellement les hypothèses de normalité en utilisant le **QQ-plot** et d'équidistribution des résidus. Ils permettent également d'évaluer la bonne adéquation des modèles ajustés. Cette analyse graphique des résidus vise à détecter d'éventuelles points atypiques : les représentations graphiques des résidus en fonction des observations, des valeurs prévues par le modèle, ne doivent faire apparaître aucune structure non aléatoire. Dans le cas contraire, un diagnostic de non-conformité aux hypothèses peut être porté.

Hypothèse d'homoscédasticité : dans le cas ou la variance des résidus de la régression est la même pour chaque observation, on parle alors d'homoscédasticité. On peut vérifier cette hypothèse graphiquement, en traçant la variance des résidus en fonction des observations.

▷ **Loi de Poisson surdispersée**

Loi de Poisson surdispersée (ou quasi-Poisson) diffère de la loi de Poisson puisque sa variance n'est pas égale mais proportionnelle à sa moyenne : Soient μ, ϕ deux réels strictement positifs. Une variable aléatoire X suit la loi de Poisson surdispersée de paramètre (μ, ϕ) si et seulement si $\frac{X}{\phi}$ suit une loi de Poisson de paramètre $\frac{\mu}{\phi}$. Dans ce cas, nous notons $X \sim \text{ODP}(\mu, \phi)$ avec ODP est l'abréviation de *Over-Dispersed Poisson*. Dans le cas de la loi de Poisson, on a $\phi = 1$.

Avec le paramètre ϕ en plus, une loi de Poisson surdispersée généralise une loi de Poisson habituelle. Une contrainte forte pour la loi de Poisson est que l'espérance et la variance sont égales. Si cette hypothèse n'est pas vérifée par les données dans le GLM, on peut considérer la distribution de Poisson surdispersée . Elle permet ainsi une relation plus flexible entre la variance et l'espérance de la variable. Concrètement, $\text{Var}[X] = \phi\mathbb{E}[X]$. Par ailleurs, la famille de loi de Poisson surdispersée possède une propriété intéressante : elle est invariante par l'additivité. Sa fonction lien canonique $g(.)$ dans le modèle GLM est la même que celle d'une loi Poisson, la fonction *log*.

▷ **Le modèle GLM de la loi de Poisson surdispersée**

• **Principe**

L'idée principale de Renshaw et Verrall est de capturer et détecter la tendance des règlements incrémentaux au fil des années de survenance et des années de développement en supposant que les règlements suivent une loi de Poisson surdispersée. Il s'agit du modèle GLM Poisson surdispersée où les variables à expliquer sont les règlements incrémentés X_{ij} et les variables explicatives les indicatrices $\mathbf{1}_k(i)$ et $\mathbf{1}_k(j)$, avec i représente l'année (ou trimestre) de survenance et j représente l'année (ou trimestre) de developpement. Ce modèle se repose sur les hypothèses suivantes :

- ***Hypothèse 1*** : Les règlements incrémentés $(X_{i,j})_{i,j}$ sont indépendants.
- ***Hypothèse 2*** : $X_{i,j}$ suit la loi de Poisson surdipersée de paramètre $(\mu_{i,j}, \phi)$ avec $\phi > 0$.
- ***Hypothèse 3*** : Il existe des paramètres réels $\alpha_1, \alpha_2, \ldots, \alpha_I$ et $\beta_1, \beta_2, \ldots, \beta_J$ tels que
$$\mu_{i,j} = \exp(\alpha_i + \beta_j).$$
- ***Hypothèse 4*** : $\alpha_1 = 0$ et $\beta_1 = 0$.

Il peut sembler curieux de modéliser des montants par une loi de Poisson, qui ne charge que les valeurs entières. En fait, Renshaw et Verrall montrent que sous l'hypothèse $\sum_{i=1}^{n-j+1} X_{ij} \geq 0$, pour tout j, l'estimation des paramètres par vraisemblance conditionnelle et la prédiction des valeurs futures *conduisent aux mêmes formules que la méhode de Chain Ladder*, ce qui légitime le fait d'utiliser cette approche, même pour des tableaux à valeurs non entières.
Les estimations des règlements incrémentés et des provisions futures sont données par
$$\hat{X}_{i,j} = \hat{\mathbb{E}}(X_{i,j}) = e^{\hat{\alpha}_i + \hat{\beta}_j} \text{ et } \hat{R} = \sum_{i=2}^{n} \sum_{j=n-i+2}^{n} \hat{X}_{i,j},$$

ainsi que l'estimation des variances sont données par

$$\hat{\mathbb{V}}\left(X_{i,j}\right) = \hat{\phi} e^{\hat{\alpha}_i + \hat{\beta}_j} \text{ et } \hat{\mathbb{V}}(R) = \sum_{i=2}^{I} \sum_{j=J-i+2}^{n} \hat{\mathbb{V}}\left(X_{i,j}\right).$$

Par additivité de la loi de Poisson surdispersé, nous obtenons sous l'hypothèse d'indépendance des incréments que les provisions totales R suivent une loi de Poisson surdispersé de paramètres $\left(\sum_{i=2}^{n} \sum_{j=n-i+2}^{n} \mu_{i,j}, \phi\right)$. Le principal inconvénient de ce modèle réside dans le fait qu'ils requièrent que les montants incrémentaux $X_{i,j}$ soint positifs. En pratique, cette contrainte est rarement vérifiée, en raison de :

▷ La présence de boni de liquidation dans le déroulement de la charge des sinistres.

▷ L'encaissement de recours en fin de développement, pour les triangles de paiements.

L'application de ce modèle nécessite donc en pratique l'utilisation de triangles de paiements bruts de recours encaissés, ou translater les données d'origine. Nous pourrions utiliser les estimations des paramètres, déterminer un quantile pour la provision totale R. Néanmoins, il faut tenir compte aussi de l'erreur d'estimation de R et non seulement de sa variance afin de quantifier la volatilité totale.

• Estimation de la volatilité des provisions

Comme nous l'avons dans le modèle de Mack, l'erreur de prédiction se mesure par *la mean square error prediction* (MSEP)

$$\text{MSEP}(\hat{R}) = \mathbb{E}\left[(\hat{R} - R)^2 \mid D_t\right],$$

où $D_t = \{X_{i,j}, i + j \leq t, 1 \leq j \leq J\}$ est le triangle supérieur. Le MSEP peut se décomposer par approximation en deux parties

$$\text{MSEP}(\hat{R}) \simeq \mathbb{E}\left[(\hat{R} - \mathbb{E}[R])^2 \mid D_t\right] + \mathbb{E}\left[(R - \mathbb{E}[R])^2 \mid D_t\right],$$

où le premier terme est l'erreur d'estimation, car $\hat{R}$ a été estimé à partir du triangle supérieur, et le second terme est la variance classique de la variable R, appelée process variance. On peut approcher l'erreur d'estimation par la variance de $\hat{R}$, d'où la formule

$$\text{MSEP}(\hat{R}) \simeq \mathbb{V}(\hat{R}) + \mathbb{V}(R).$$

Calculer l'erreur d'estimation est moins évident. Une façon de l'estimer alors c'est d'utiliser une méthode de Bootstrap.

1.6.3.3 Méthode de ré-échantillonnage (Bootstrap)

▷ Principe

Le bootstrap (une vision non paramétrique) est une technique de ré-échantillonnage simple et puissante, permettant d'obtenir à partir d'un unique échantillon de données plusieurs informations pour estimer la distribution des provisions. Son principe est de simuler de nouvelles données similaires aux données observées, à partir desquelles on déduit un échantillon de réalisations de $\hat{R}$.

Les données sont supposées indépendantes et identiquement distribuées (i.i.d) dans l'application classique du ré-échantillonnage. En revanche, dans le cadre du modèle GLM Poisson surdispersée, les variables sont supposées indépendantes mais pas identiquement distribuées. Par conséquent, la méthode du ré-échantilllonnage sera appliquée non pas directement aux variables mais aux résidus de

Pearson qui sont plus susceptibles de posséder cette propriété. On peut alors calculer une variance empirique comme estimateur de $\mathbb{V}(\hat{R})$. La méthode de Boostrap repose sur l'hypothèse suivante : Les variables rééchantillonnées doivent être *indépendantes et identiquement distribuées* (*iid*) : recours aux résidus.

La pratique du bootstrap dans le cas du modèle GLM Poisson surdispersée, se fait par l'algorithme suivante :

1. Estimation initiale des paramètres sur le triangle supérieur.
2. Déduction d'un triangle supérieur d'incréments moyens $\hat{\mu}_{i,j}$.
3. Calcul des résidus de Pearson $r_{i,j} = \frac{X_{i,j} - \hat{\mu}_{i,j}}{\sqrt{\hat{\mu}_{i,j}}}$.
4. Itération M fois : pour $k = 1, \ldots, M$:
   (a) Re-échantillonnage des résidus et obtention d'un nouveau triangle supérieur de résidus $r_{i,j}^{(k)}$.
   (b) Calcul d'un nouveau triangle supérieur d'incréments $X_{i,j}^{(k)} = \hat{\mu}_{i,j} + r_{i,j}^{(k)} \sqrt{\hat{\mu}_{i,j}}$. Dans le cas où les incréments obtenus sont négatifs, le triangle est rejeté et un nouveau re-échantillonnage des résidus est réalisé.
   (c) Utilisation du modèle pour re-estimer les paramètres et calcul d'une nouvelle provision totale $R^{(k)}$.
5. Estimation de $\mathbb{V}(\hat{R})$ empiriquement à partir de l'échantillon $\left\{R^{(k)}, k = 1, \ldots, M\right\}$.

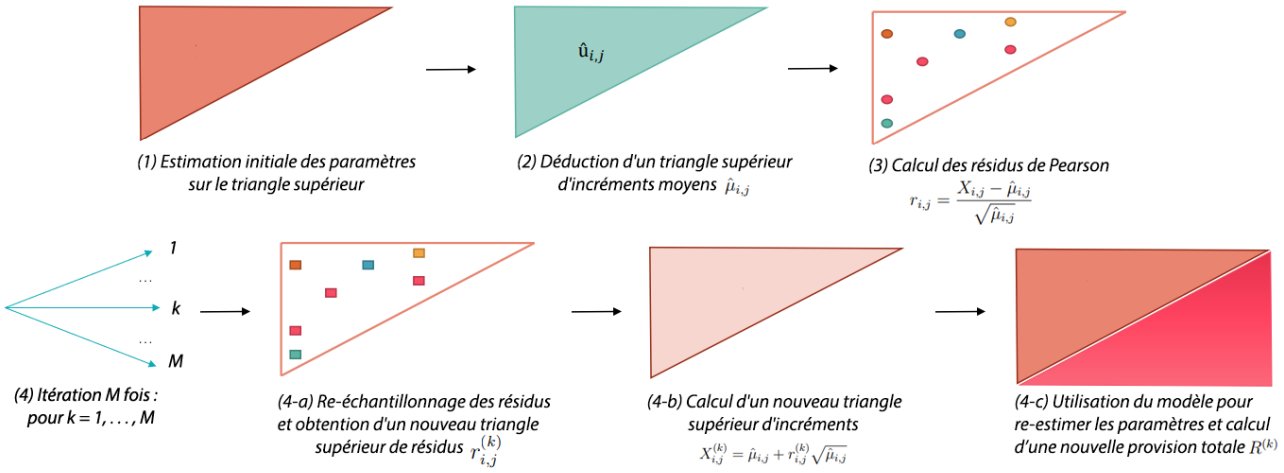

FIGURE 1.9 : Synthétise les différentes étapes de l'algorithme Bootstrap.

Cette méthode est dite semi-simulatoire, parce que la MSEP est obtenue en additionnant la variance de R (process variance) et la variance de $\hat{R}$ (erreur d'estimation), obtenue par simulation. Alors une fois obtenues les estimations de la provision totale moyenne et de la volatilité totale, il est donc possible de calculer un quantile, en choisissant de façon usuelle comme loi de référence une loi lognormale ou normale.

▷ Validation du modèle

Afin de valider la méthode de bootstrap, il faut alors que la loi bootstrap converge faiblement, en probabilité, vers la loi réelle. Cependant, pour comparer les deux distributions, la distance (métrique) de Mallows permet de montrer que la distance entre la loi réelle et la loi bootstrap converge vers zéro quand le nombre d'observations tend vers l'infini.

Chapitre 2

Méthodes de Machine Learning et de Deep Learning en provisionnement

2.1 L'évolution du Machine Learning et du Deep Learning

2.1.1 Machine Learning (ML)

• Introduction

L'Intelligence Artificielle (IA) est l'ensemble des techniques et théories qui cherchent à développer des modèles capables de simuler le comportement humain. Parmi ces techniques, on trouve le Machine Learning (ML), très populaire depuis 2010. Le Deep Learning (DL) est un domaine du Machine Learning qui est focalisé sur le développement des réseaux de neurones et qui fait face à d'autres défis que ceux du machine Learning. Parmi ces défis : comment entrainer des modèles avec une infinité de paramètres et de données dans des temps raisonnables.

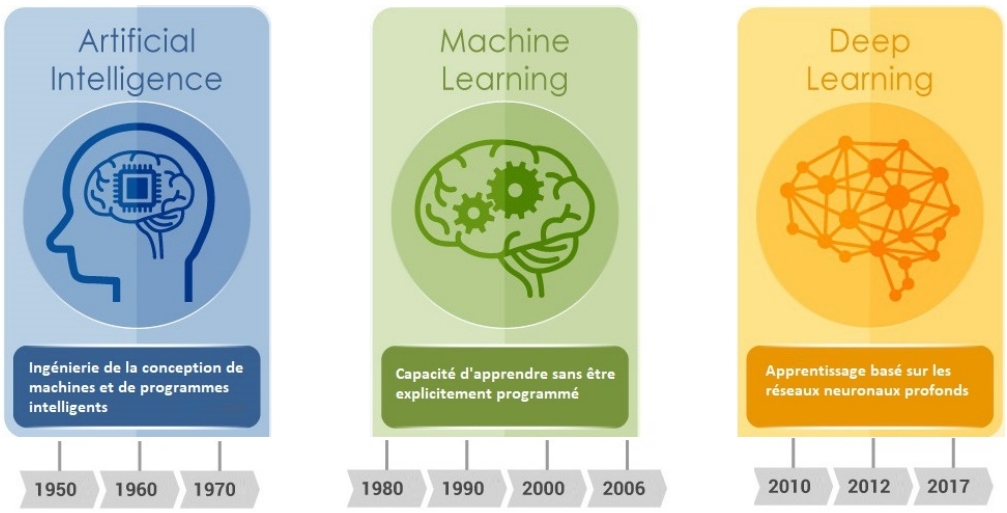

FIGURE 2.1 : L'évolution de l'Intelligence Artificielle (*source : Amah www.geekmaispasque.com*).

En Machine Learning on créé un programme qui acquiert une aptitude : laisser l'ordinateur apprendre quel calcul effectuer, alors qu'en Data Science, on analyse plus souvent des données pour en créer un modèle en réaction à ces données. Il existe trois méthodes d'apprentissage : l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement. On s'interesse par la suite à l'apprentissage supervisé.

• **L'Apprentissage Supervisé**

L'Apprentissage Supervisé est utilisé afin de développer des modèles prédictifs, capables de prédire une valeur cible y (target) en fonction de variables $x_1, x_2, ..., x_n$ (features).

• **Le modèle et ses paramètres**

Un modèle est une fonction mathématique développée à partir d'une base de données. Il associe une variable d'entrée x à une variable de sortie y telle que $y = f(x) + \epsilon$, avec ϵ represente l'erreur. Un bon modèle doit être une bonne généralisation, il doit fournir de petites erreurs, sans être sujet au sur-apprentissage (*l'Over-fitting*), c'est à dire que le modèle soit trop adapté aux données d'apprentissage et ne se généralise pas à de nouvelles données qui lui sont inconnues.

• **Fonction coût**

La fonction coût mesure l'ensemble des erreurs entre le modèle et le jeu de données. Pour avoir un meilleur modèle, il faut minimiser cette fonction. Dans le cas de régression, on utilise souvent les deux fonctions coût (indicateurs de performance) **RMSE** et **MAE** :

RMSE (Root Mean squar error) est la racine carrée de la moyenne arithmétique des carrés des écarts entre les prédictions $\hat{y}_i$ et les observations y_i :

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

MAE (Mean absolute error) est la moyenne arithmétique des valeurs absolues des écarts entre les valeurs observées y_i et les valeurs prédites $\hat{y}_i$:

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

• **Algorithme d'apprentissage**

L'idée pricipale du Machine Learning, c'est de laisser la machine trouver quels sont les paramètres de notre modèle qui minimisent la fonction coût. Pour cela, on utilise un algorithme d'apprentissage. tel que l'algorithme de **descente de gradient** (gradient descent) qui est un algorithme itératif. Il existe beaucoup de variante de cet algorithme, on trouve par exemple : *Mini Batch Gradient Descent*, *Stochastic Gradient Descent*, *Momentum*, *RMSProp* et *Adam*.

• **Régression et classification**

Les modèles sont développées pour deux types de problèmes : les problèmes de *Régression* dans le cas où on veut prédire la valeur de y qui est une variable quantitative et les problèmes de *Classification* dans le cas où on veut prédire la valeur d'une variable qualitative.

• **Data pre-processing**

La performance d'un modèle de Machine Learning dépend fortement de la quantité de données utilisées. Par contre, avoir beaucoup de données ne suffit pas, il faut aussi avoir de bonnes données et bien les comprendre. Alors, l'étape de préparation des données (*Data pre-processing*) est très importante, donc avant de commencer à appliquer les modèles, il est nécessaire de procéder à quelques modifications sur la base de données :

◦ Supprimer les anomalies et les erreurs pour ne pas biaiser l'apprentissage.

◦ Normaliser les données pour rendre l'apprentissage rapide et efficace.

- Completer ou supprimer les données manquantes.
- Binariser les variables catégoriales pour un apprentissage afficace.
- Analyser la corrélation entres les variables.
- Créer de nouvelles variables en se basant sur les autres variables (*feature engineering*).

• **Sélection et évaluation du modèle**

Lors de la sélection d'un modèle, nous distinguons 3 partitions différentes des données :

Échantillon d'apprentissage (Training set) : Jeu de données utilisé pour entraîner un algorithme, qui représente généralement 80% de la base de données.

Échantillon de validation (Validation set or hold-out) : Jeu de données utilisé pour valider les paramètres de l'algorithme, qui représente généralement 20% de la base de données.

Échantillon de test (Testing set set) : Jeu de données utilisé pour tester les performances de l'algorithme. Parfois il remplace l'échantillion de validation.

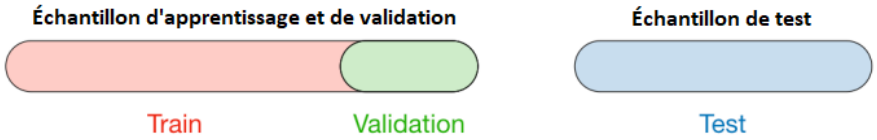

FIGURE 2.2 : Répartition de la partition d'apprentissage, de validation et de test.

On peut aussi utiliser la validation croisée ou **Cross Validation (CV)**, une méthode qui permet de sélectionner et évaluer les performances des modèles. Il existe deux types de validation croisée : k-fold et Leave-p-out.

k-fold : Entraînement sur $k-1$ jeux de données et évaluation sur les autres jeux, par défaut $k=5$ ou $k=10$.

Leave-p-out : Entraînement sur $n-p$ observations et évaluation sur les p restantes.

La méthode la plus couramment utilisée est appelée " *Validation Croisée k-folds*" qui sépare les données d'apprentissage en k partitions pour valider le modèle sur une seule, tout en entraînant le modèle sur les $k-1$ autres partitions, en itérant cette procédure k fois. L'erreur est alors moyennée sur les k partitions et est appelée Erreur de Validation Croisée.

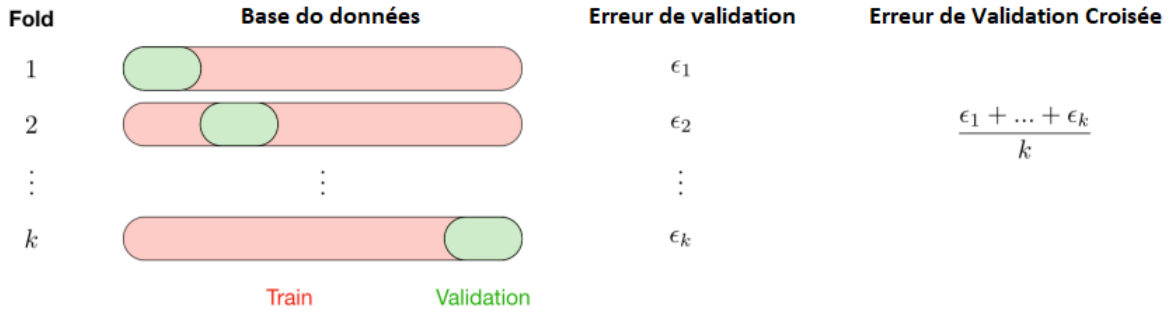

FIGURE 2.3 : Présentation de la Validation Croisée k-fold.

• **Modèles usuelles de Machine Learning**

Les modèles les plus utilisés en Machine Learning sont :

⋆ Arbres de décision (**Decision trees**) .
⋆ Les Forêts Aléatoires (**Random Forest**).
⋆ Le Gradient Boosting (**XGBoost**).
⋆ Les Réseaux de Neurones (**Neural Networks**) en *Deep Learning*.

2.1.2 Deep Learning (DL)

Le Deep Learning (l'apprentissage profond) est une sous-partie du Machine Learning. Il est l'ensemble de méthodes d'apprentissage qui visent à modéliser les données à l'aide d'architectures très complexes combinant différentes transformations non-linéaires. Ses composantes élémentaires sont les réseaux de neurones artificiels, qui sont combinés pour former les réseaux de neurones profonds. Ces techniques ont notamment permis des progrès significatifs dans des domaines différents. Les applications potentielles sont très nombreuses, l'exemple le plus impressionnant est le programme AlphaGo, qui a appris à jouer au jeu de go par la méthode du Deep Learning, et a battu en 2016 le champion du monde !

Les algorithmes du Deep Learning (réseaux de neurones) cherchent à tirer des conclusions similaires à celles des humains en analysant de manière continue les données avec une structure logique donnée. Pour y parvenir, le Deep Learning utilise une structure multi-couches.

Les différentes couches des réseaux neuronaux peuvent également être considérées comme une sorte de filtre, ce qui augmente la probabilité de détecter et de produire un résultat correct. En général, les réseaux de neurones artificiels possèdent des capacités uniques qui permettent aux modèles de Deep Learning de résoudre des tâches que les modèles de Machine Learning ne pourront jamais résoudre.

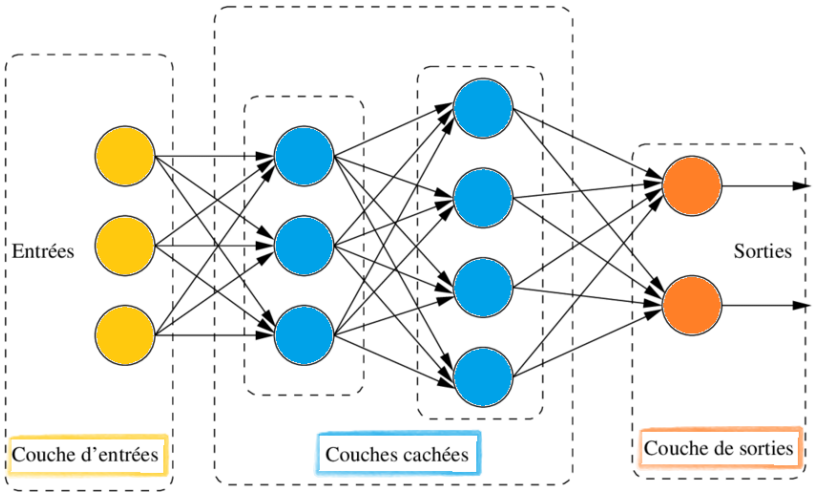

FIGURE 2.4 : Réseaux de neurones multi-couches.

• **Les avantages de Deep Learning**

⋆ **Pas de Feature Extraction :**

Les algorithmes de Machine Learning sont appelés "***Flat Algorithms***". Le terme "flat" fait référence au fait que ces algorithmes ne peuvent normalement pas être appliqués directement aux données brutes. Nous avons besoin d'une étape de prétraitement appelée "***Feature Extraction***", c'est-à-dire de faire un choix, et extraire la donnée qui va influer sur la prédiction. Les réseaux neuronaux artificiels n'ont pas besoin de l'étape *Feature Extraction*. Les couches sont capables d'apprendre directement et par elles-mêmes une représentation implicite des données brutes.

⋆ **S'adapter au le Big Data :**

Le deuxième grand avantage du Deep Learning (qui explique en grande partie pourquoi il devient si populaire) c'est qu'il est alimenté par des quantités massives de données. Le Big Data offrira d'énormes possibilités d'innovations en matière de Deep Learning. Par contre, les méthodes de Machine Learning ne s'adapte pas vraiment dans le cas des données massives.

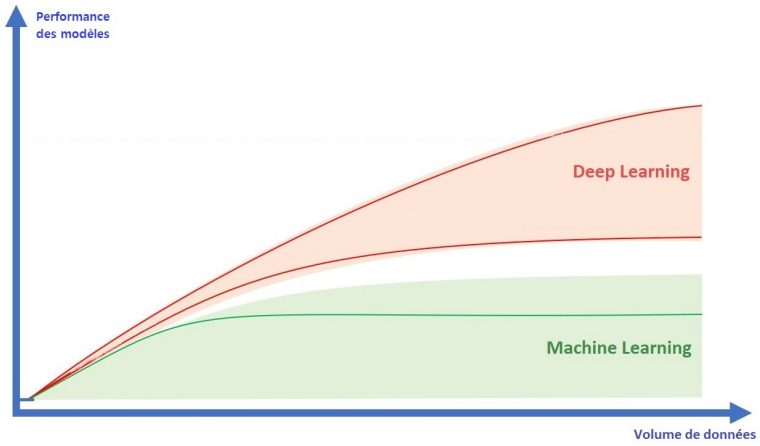

FIGURE 2.5 : Les performances de Machine et Deep Learning en fonction du volume de données.

Les modèles de Deep Learning ont tendance à accroître leur performances avec l'augmentation de la quantité de données d'apprentissage, alors que les modèles de Machine Learning cessent de s'améliorer après un point de saturation.

• **Les limites de Deep Learning**

⋆ **Puissance de calcul :**

Pour avoir un algorithme de Deep Learning performant, il faut avoir un algorithme très profond. Cela nécessite une puissance calculatoire importante, il faut donc avoir des ordinateurs plus performants, sinon l'algorithme va prendre un temps énorme pendant la phase d'apprentissage.

⋆ **Une interprétabilité difficile :**

L'un des principaux inconvénients de Deep Learning est sa logique interne pour atteindre la sortie ou le résultat souhaité, qui est incompréhensible et non intérpretable. Ce comportement du réseau neuronal profond est connu sous le nom de "***Black Box***" (boîte noire).

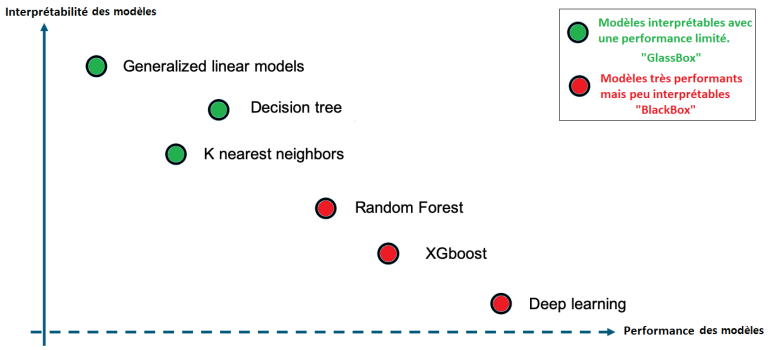

FIGURE 2.6 : Representation de l'interprétabilité des modèles de ML et DL en focntion du performance.

2.1.3 Le succès des réseaux de neurones en provisionnement

L'intérêt croissant pour les réseaux de neurones est associé à un nombre plus important de publications au cours des cinq dernières années. Ces publications marquent un renforcement de la volonté de recourir à des méthodes dont l'utilisation est encore peu répandue dans le domaine du provisionnement et aussi de tirer parti des avancées en Deep Learning pour proposer de nouveaux modèles de provisionnement en assurance non-vie afin de concurrencer les méthodes usuelles. Parmi ces publications, on trouve :

- **Réseaux neuronaux appliqués à Chain-Ladder, Wüthrich (2018)**

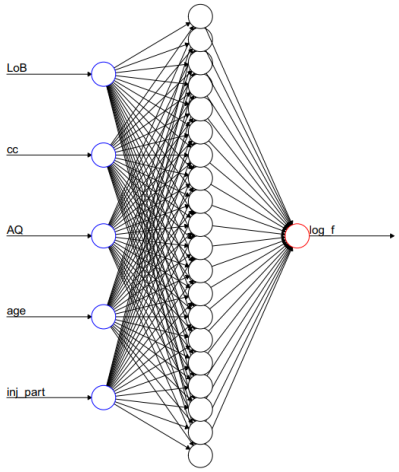

FIGURE 2.7 : Le modèle réseaux neuronaux appliqués à Chain-Ladder de WÜTHRICH (2018).

En 2017, WÜTHRICH (2018) a proposé pour la première fois, un réseau de neurones qui prend certaines caractéristiques des triangles de développement pour prédire les facteurs de développement de Chain Ladder. Le modèle se compose d'un réseau neuronal avec une couche cachée de 20 neurones.

- **DeepTriangle : Une approche de Deep Learning, Kuo (2019)**

En 2019, en réponse au programme d'appel à contributions sur les provisions du CAS (Casualty Actuarial Society), KUO (2019) a proposé une architecture de réseau de neurones récurrent (GRU) pour produire des séquences de paiements futurs et les montants des sinistres en cours depuis un triangle de développement.

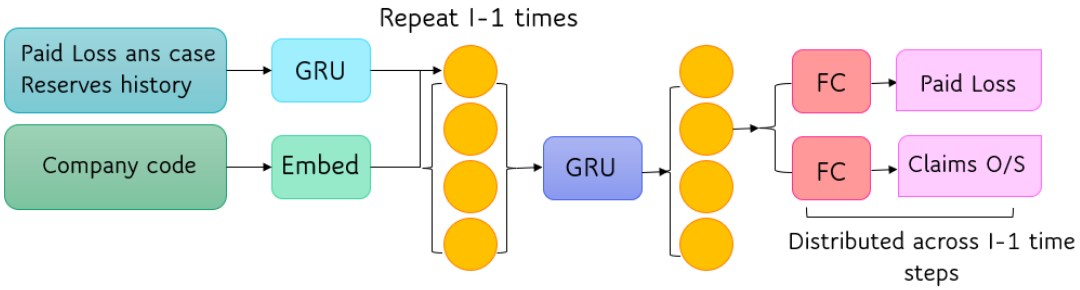

FIGURE 2.8 : Le modèle DeepTriangle de KUO (2019).

- **Neural network boosted double overdispersed de Gabrielli (2019)**

GABRIELLI (2019), un des étudiants de Mario Wuthrich, a proposé le modèle "*Neural Network boosted ODP (overdispersed Poisson)*". Son principe est d'examiner les prédictions du Modèle GLM Poisson surdispersé (ODP) à partir d'un triangle de développement et former un réseau de neurones.

Le réseau de neurones s'initialise pour donner les prédictions du modèle ODP, entraîné sur les résidus, c'est donc une sorte de processus de boosting.

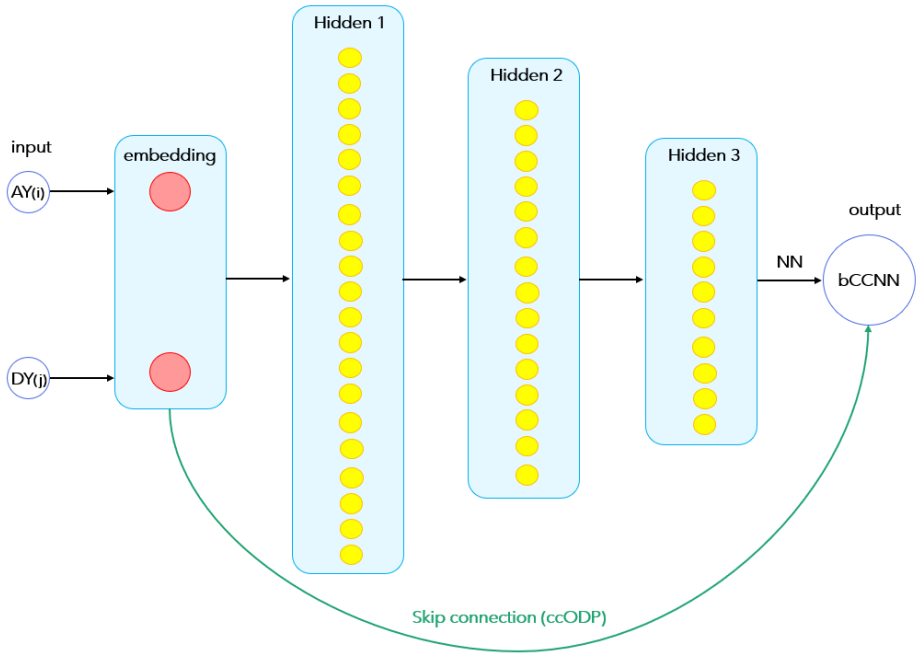

FIGURE 2.9 : Le modèle Neural Network boosted overdispersed de GABRIELLI (2019).

Dans le cadre du suivi de ces travaux, il a également incorporé le nombre de sinistres en plus des montants des sinistres, puis a effectué ce que nous appelons un "Apprentissage Multitâche" dans le réseau de neurones pour prédire les deux quantités simultanément avec une meilleure précision.

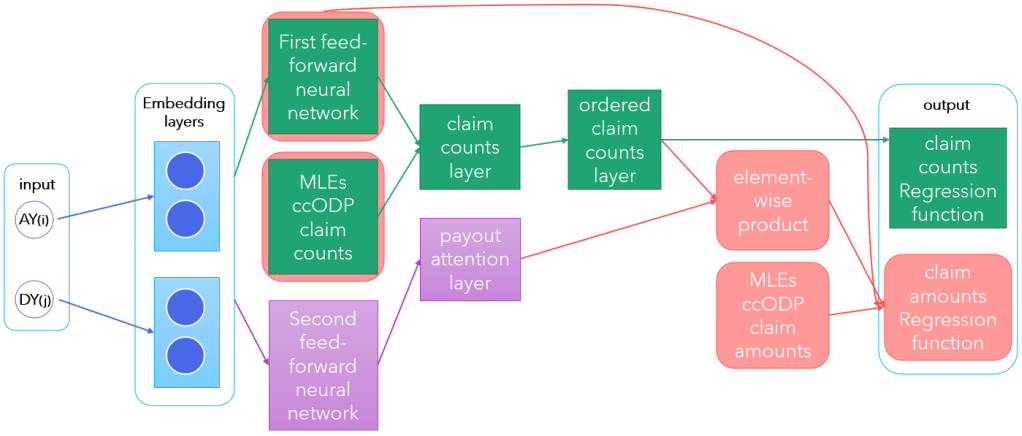

FIGURE 2.10 : Le modèle Neural network boosted double overdispersed de GABRIELLI (2020b).

• **Provisionnement stochastique avec des réseaux de densité mélange ResMDN,M. Taher Al-Mudafer (2021)**

En s'inspirant des travaux de WUTHRICH (2019) et GABRIELLI (2020b), Muhammed Al-Mudafer en collaboration avec Benjamin Avanzi, Greg Taylor et Bernard Wong, a developpé en 2021 un modèle ResMDN en se basant sur les deux modèles suivants : GLM Poisson surdispersé (ODP) et réseau de densité mélange (MDN). Le modèle ODP forme la base pour une interprétabilité facile, tandis que le MDN renforce les résidus du ODP et détecte les tendances que le GLM a négligées. Cependant, le

modèle ResMDN est plus interprétable, mais moins performant que le modèle MDN.

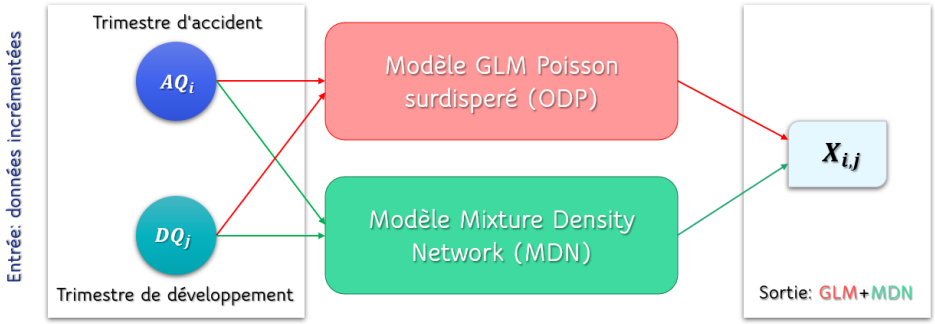

FIGURE 2.11 : Le modèle ResMDN (GLM-MDN) de M. TAHER AL-MUDAFER (2021).

2.1.4 Méthode de validation des modèles

La majorité de la littérature sur les méthodes de Machine Learning et de Deep Learning appliquées au provisionnement non-vie utilise le triangle de développement supérieur uniquement pour l'apprentissage et la validation des algorithmes. En effet, il n'existe pas de répartition complete du triangle de liquidation en échantillion d'apprentissage, de validation et de test. Par conséquent, il n'existe pas de cadre permettant de tester et de sélectionner les différents modèles sur le triangle supérieur. La mise en œuvre des modèles de Machine Learning et de Deep Learning dans la pratique nécessitera un cadre qui partitionne le triangle supérieur et permet de tester hors échantillon différents modèles.

Plusieurs méthodes de validation de modèles notables ont été utilisées récemment. En 2017, WÜTHRICH (2018) a effectué une répartition aléatoire (90% pour l'apprentissage et 10% pour la validation) sur le triangle supérieur. Cette répartition risque de rendre les projections instables, car le modèle formé n'a pas été testé sur les données des années calendaires futures. En 2019, KUO (2019) limite ce risque en utilisant les dernières années calendaires du triangle pour la validation. En 2020, BALONA et RICHMAN (2020) effectuent un partitionnement séquentiel du triangle de charges de sinistres. Par contre, l'utilisation de données de test à l'intérieur du triangle supérieur qui n'ont pas été vues pendant la phase d'apprentissage fournira une mesure plus fiable de la précision de la projection.

2.1.4.1 La méthode ROCV (Rolling-Origin Cross-Validation)

Le triangle de liquidation peut être considéré comme un ensemble de séries temporelles, une pour chaque période d'accident. L'étude des méthodologies de prédiction des séries temporelles aidera à trouver des moyens de tester différents modèles de Machine Learning et de Deep Learning.

La validation croisée (CV) mesure le pouvoir prédictif d'un modèle sur des données non observées. En 2012, BERGMEIR et BENTEZ (2012) soulignent que la validité de cette méthode s'effondre pour l'analyse des séries temporelles, en raison des dépendances temporelles au sein de la séquence de données et de l'absence de prévisions futures. En 2020, YANG (2020) a précisé que la séparation des données doit se faire dans l'ordre chronologique, c'est-à-dire que les données d'apprentissage précèdent les données de validation et les données de validation précèdent les données de test. Ensuite, il a analysé les techniques d'évaluation des modèles à origine fixe et à *origine roulante* (*Rolling-Origin*), couramment utilisées dans l'analyse des séries chronologiques.

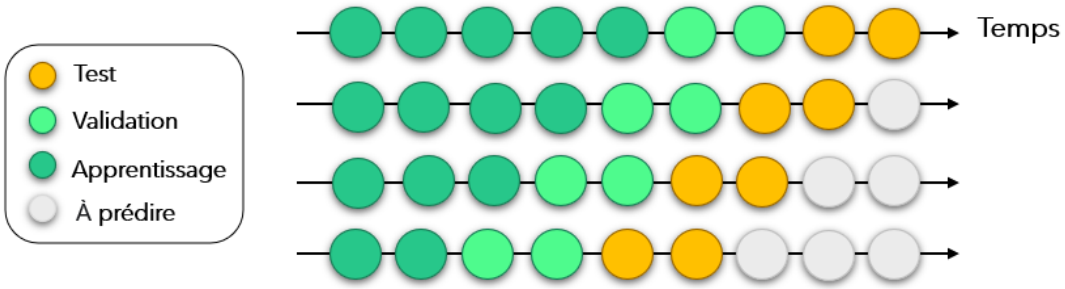

FIGURE 2.12 : Représentation de la méthode ROCV.

La méthode de l'origine fixe suppose une partition fixe d'apprentissage, de validation et de test, effectuée dans l'ordre chronologique. L'origine roulante est analogue à la méthode de validation croisée. Elle implique une partition chronologique des données, similaire à la méthode d'origine fixe.

La différence réside dans le fait que plusieurs partitions sont effectuées : la partition initiale utilise uniquement la partie antérieure des données, dans les partitions suivantes, les données d'apprentissage sont progressivement étendues jusqu'à ce que toutes les données soient utilisées. En 2020, BALONA et RICHMAN (2020) ont tout récemment appliqué la méthodologie ROCV au triangle de développement, démontrant ainsi son efficacité dans la sélection de modèles. Bien que la méthode à origine fixe soit plus simple à mettre en œuvre, elle nécessitera une grande période de prévision pour être réalisable, en particulier dans le cadre des triangles de développement où la période de prévision est égale à la taille des données disponibles. Une partition plus importante de l'ensemble de test réduira les données d'apprentissage utilisées pour le test, ce qui conduira à des prévisions plus volatiles.

Dans notre étude, on va utiliser la méthode *ROCV* avec une validation croisée *5-fold* afin de réglage les hyperparamètres des modèles de Machine Learning et de Deep Learning, ainsi de maximiser l'utilisation des données d'apprentissage et de test, les premières partitions utilisant davantage de données de test pour évaluer la précision de projection du modèle, tandis que les partitions ultérieures évaluent la capacité du modèle à à capturer les tendances dans l'ensemble de données.

2.1.4.2 Évaluation de la précision de la projection

Lorsque l'objectif de la modélisation est d'améliorer la précision de l'interpolation, la répartition aléatoire des données en ensembles d'apprentissage, de validation et de test est souvent suffisante. Pour les triangles de développement, l'objectif est l'extrapolation, et l'ensemble de test doit donc se concentrer sur l'évaluation de la précision de projection du modèle. Pour ce faire, on affecte les dernières périodes calendaires du triangle à l'ensemble de test et les plus anciennes à l'ensemble d'apprentissage. De même, la partition de validation est choisi pour être les dernières périodes calendaires qui ne sont pas affectées au test. De cette façon, les modèles arrêtent la phase d'apprentissage lorsque la précision de la projection à court terme est maximisée. Le processus de modélisation se déroulera en 4 étapes :

▷ Pour la première étape, les données de test sont composées d'un plus grand nombre de périodes calendaires ultérieures. ***Cette étape se concentrera sur l'évaluation de la précision des modèles et le réglage des hyper-paramètres*** lors de la projection dans le Triangle inférieur. L'inconvénient est la réduction des données d'entraînement, d'autant plus que les périodes calendaires ultérieures contiennent des informations précieuses sur les tendances futures des sinistres.

▷ La deuxième étape fournira davantage de périodes calendaires pour l'ensemble d'apprentissage. Cette étape permettra d'évaluer la précision des modèles et le réglage des hyper-paramètres lorsqu'ils utilisent la quasi-totalité du triangle et ***la façon dont ils capturent les tendances dans l'ensemble des données***.

▷ La troixième étape permettra l'apprentissage et la validation du modèle final, *après avoir sélectionner les meilleurs hyper-paramètres qui minimisent les erreurs dans l'étape 1 et l'étape 2 à la fois*, en utilisant *ROCV* avec *5-fold*. En revanche, la quatrième étape permettra la projection et la prédiction du triangle inférieur.

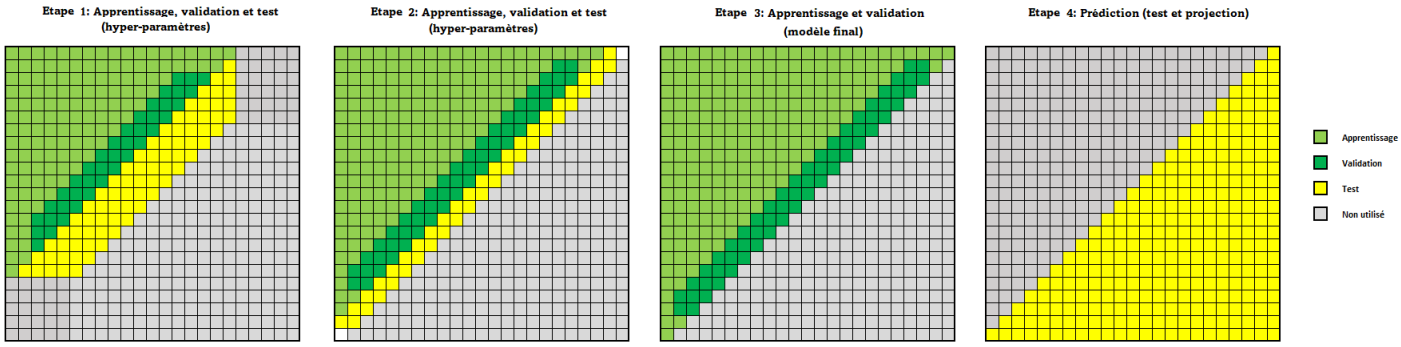

FIGURE 2.13 : Les 4 étapes du processus de modélisation.

2.2 Méthodes de Machine Learning en provisionnement

Parmi les méthodes performants en Machine Learning, on trouve : ***Decision tree***, ***Random Forest*** et ***XGBoost***. L'application de ces méthodes en provisionnement non-vie sera interessant afin de comparer les résultats obtenus avec ceux de Deep Learning et des méthodes usuelles.

2.2.1 Arbre de décision (Decision Tree)

Les arbres de décision (Decision Tree) sont des algorithmes de Machine Learning utilisées dans les problèmes de régression et de classification. Leur principe repose sur l'algorithme *CART* (BREIMAN (1996)) dans le but de construire une classification hiérarchique descendante des observations en des catégories homogènes par rapport à la variable à prédire, c'est à dire de trouver une partition qui sépare au mieux les différentes observations. Une fois segmenté, il crée un ensemble de règles appelé *séquences de décision uniques par groupe* en vue de la prédiction d'un résultat ou d'une classe.
L'ensemble des noeuds se divise en trois catégories :

Noeud racine : ce noeud permet l'accès à l'arbre.

Noeuds internes : ces noeuds ont des descendants.

Feuilles (noeuds terminaux) : ces noeuds n'ont pas de descendants.

A chaque noeud de l'arbre, une variable est choisie pour séparer la classe initiale en deux sous-classes. Les tests s'effectuent dans les noeuds internes, et les décisions sont prises dans les noeuds terminaux (feuilles).

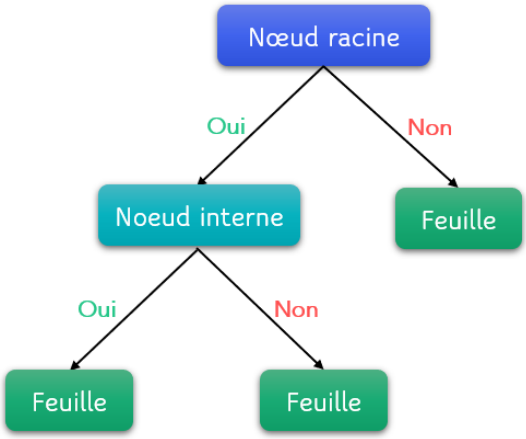

FIGURE 2.14 : Représentation graphique d'un arbre de décision.

Les deux principaux critères de séparation des classes sont ***la variance intra-noeuds*** et ***le critère de Gini***. En principe, ces deux critères donnent des résultats très proches bien que les formules soient différentes. Pour le critère de Gini : l'algorithme cherche à minimiser la quantité suivante à chaque noeud

$$J\left(k, t_k\right) = \frac{m_{\text{gauche}}}{m} G_{\text{gauche}} + \frac{m_{\text{droite}}}{m} G_{\text{droite}} ,$$

k **:** la variable explicative considérée.

t_k **:** le seuil considéré.

m **:** le nombre d'instances dans l'échantillon initial.

$m_{\textbf{gauche/droite}}$ **:** le nombre d'instances dans le noeud gauche/droite.

$G_{\textbf{gauche/droite}}$ **:** l'indice de Gini du noeud gauche/droite.

Pour un noeud considéré, on obtient l'indice de Gini par la formule suivante

$$G = 1 - \sum_k p_k^2,$$

avec p_k est la proportion des instances de la classe k dans le noeud considéré. L'indice de Gini vaut alors 0 dans le cas où un noeud ne contient que des instances d'une même classe. L'algorithme s'arrête quand chaque feuille ne contient plus qu'une seule instance, dans ce cas on parle d'arbre saturé.

- **Construction de l'arbre**

Chaque noeud interne de l'arbre correspond à un test fait sur une des variables :

Variable quantitative : test par intervalles (tranches) de valeurs.

Variable qualitative : génère une branche (un descendant) par valeur de l'attribut.

Au début, les valeurs d'entrées sont placés dans le nœud racine. Une des variables de description "variable cible" est la classe du point. Cette variable peut être qualitatives ou quantitatives. Chaque nœud est coupé par l'opération split, donne naissance à plusieurs nœuds descendants. Un élément des variables d'entrées situé dans un nœud se retrouvera dans un seul de ses descendants.

L'arbre est construit par partition successive de chaque noeud en fonction de la valeur de l'attribut testé à chaque itération. Le critère optimisé est l'homogénéité des descendants par rapport à la variable cible. La variable qui est testée dans un noeud sera celle qui maximise cette homogénéité. Le processus s'arrête quand les éléments d'un noeud ont la même valeur pour la variable cible.

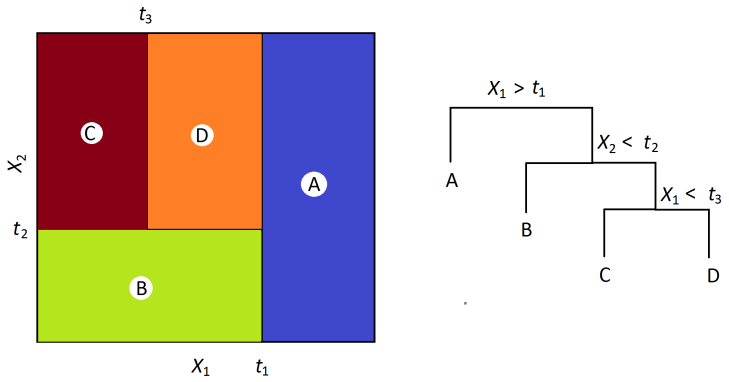

FIGURE 2.15 : Construction par partition récursive de l'espace.

Il existe plusieurs méthodes de faire des découpes. Par contre, il faut pouvoir mesurer la qualité de la découpe effectuée. On peut alors appliquer une :

Séparation par combinaison linéaire de plusieurs variables : qui donne lieu à des découpes obliques (figure 3.16 à droite).

Séparation par partition de variables : qui donne lieu à des découpes orthogonales (figure 3.16 à gauche).

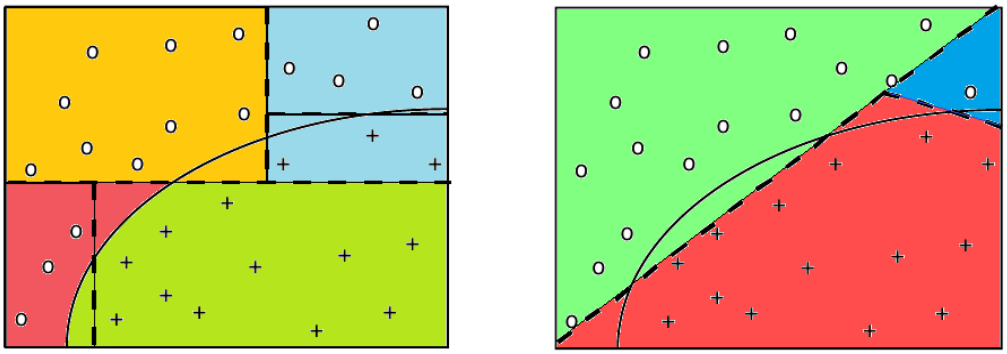

FIGURE 2.16 : Séparation de classes par partition itérative des variables.

- **Optimisation de l'arbre de décision**

Il est nécessaire, avant l'entraînement de l'arbre de décision, de fixer certains paramètres appelés *hyper-paramètre*, qui appliqueront des contraintes sur ces méthodes :

- La profondeur maximale que peut avoir l'arbre (`maxdepth`).
- Le paramètre de régularisation (`cp`).
- Le nombre d'instances minimum qu'un noeud doit avoir pour effectuer une séparation (`minsplit`).

Ces paramètres peuvent prendre une infinité de valeurs, il est donc impossible de tester toutes les combinaisons possibles. La méthode *Grid Search* permet de sélectionner les hyper-paramètres optimaux en utilisant une validation croisée.

- **Les Limites du modèle**

1) Dans certains cas, la méthode CART peut entraîner des arbres extrêmement complexes, ce qui conduit au sur-apprentissage.

2) La sensibilité des arbres de décision aux variations dans la base de données. En effet, l'apprentissage sur des bases d'entraînement différentes peut produire des arbres aux prédictions différentes.

2.2.2 Forêt aléatoire (Random Forest)

L'algorithme Forêt aléatoire (Random Forest) appartient à la famille des agrégations de modèles (Breiman (2001)). Cet algorithme utilise des stratégies adaptatives "*boosting*" ou aléatoires "*bagging*" et s'appuie sur les arbres de décisions, notamment des arbres CART, en essayant de réduire au maximum la variance de ceux-ci. L'idée principale de cet algorithme est d'utiliser une agrégation d'un grand nombre de modèles tout en évitant le sur-apprentissage.

L'idée du *bagging* c'est de créer plusieurs entités d'arbre de décision : prendre plusieurs arbres de décision et d'entrainer chacune de ces entités sur une portion aléatoire du jeu de données. Pour cela, on utilise le "*bootstrap*" qui consiste à replacer après chaque tirage aléatoire les données et les variables qui ont été sélectionnées dans notre jeu de données. De cette manière, on obtient un groupe de modèles diversifiés puisqu'ils n'ont pas tous été implémentés avec les mêmes jeux de données, mais qui partagent quand même certaines connaissances en commun, ce qui permet de réduire la variance. Une fois qu'on a ce groupe, on regroupe donc les résultats de chaque arbre de décision pour faire notre prédiction finale.

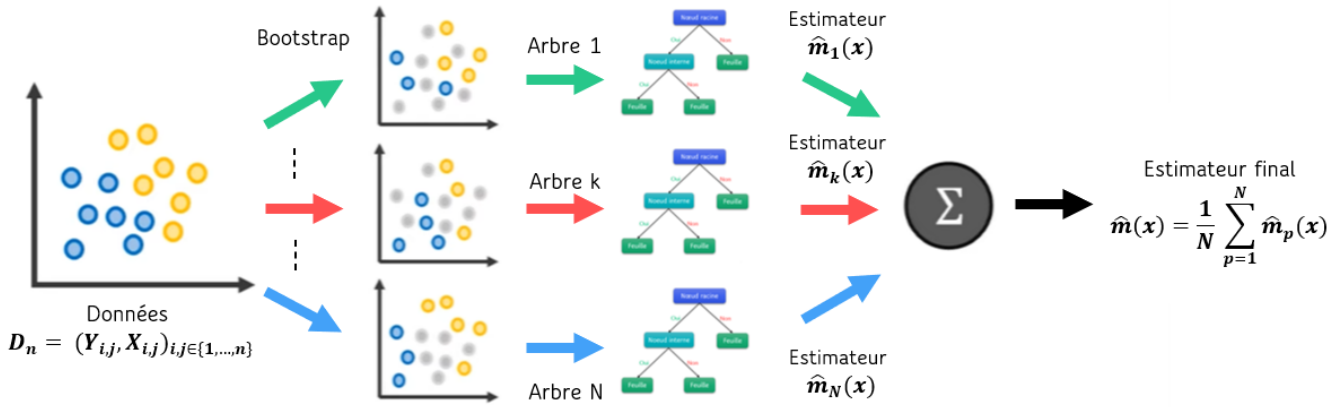

Figure 2.17 : Représentation du bagging dans le modèle forêt aléatoire.

Pour un problème de régression, on considère $(\mathbf{X}, Y)$ un vecteur aléatoire où $X \in \mathbb{R}^p$ et $Y \in \mathbb{R}$. Soit $\mathcal{D}_n = (\mathbf{X}_{i,j}, Y_{i,j})_{(i,j)\in\{1,\dots,n\}}$ notre jeu de données et $m(\mathbf{x}) = \mathbf{E}[Y \mid \mathbf{X} = \mathbf{x}]$ la fonction de régression. La méthode du bagging consiste à agréger un nombre N d'estimateurs $\hat{m}_1, \dots, \hat{m}_N$

$$\hat{Y} = \hat{m}(\mathbf{x}) = \frac{1}{B}\sum_{k=1}^{N} \hat{m}_k(\mathbf{x}).$$

Sous l'hypothèse que les régresseurs $\hat{m}_1, \dots, \hat{m}_N$ sont *i.i.d.*, avec

$$\mathbf{E}[\hat{m}(\mathbf{x})] = \mathbf{E}\left[\hat{m}_1(\mathbf{x})\right] \quad \text{et} \quad \mathbf{V}(\hat{m}(\mathbf{x})) = \frac{1}{N}\mathbf{V}\left(\hat{m}_1(\mathbf{x})\right).$$

L'estimateur agrégé $\hat{m}$ aura le même biais que les estimateurs $\hat{m}_k$ mais *avec une variance plus faible.*

- **Optimisation de forêt aléatoire**

Afin d'optimiser le modèle de forêt aléatoire par la méthode ***Grid Search***, il faut bien déterminer les hyper-paramètres du modèle CART et aussi les hyper-paramètres propres au Random Forest :

- Le nombre d'arbres créés (*ntree*).
- Nombre de variables échantillonnées aléatoirement comme candidates à chaque division (*mtry*).
- Le nombre d'instances minimum qu'une feuille doit contenir (*nodesize*).
- Nombre maximal de nœuds terminaux (*maxnodes*).

- **Les Limites du modèle**

La construction de l'algorithme de Random Forest limite le sur-apprentissage en utilisant un grand nombre d'arbres. En revanche, plus ce nombre augmente, plus le temps de calcul dans la phase d'apprentissage est important. En outre, la multiplication du nombre d'arbres réduit l'interprétabilité de l'algorithme : on parle alors de boîte noire.

2.2.3 XGBoost (eXtreme Gradient Boosting)

L'algorithme XGBoost appartient à la famille des méthodes ensemblistes qui bénéficent d'un traitement parallélisé optimisant les temps de calcul, en se basant sur la descente du gradient. Ce modèle est développé par Friedman (1999). Cette technique utilise un regroupement d'arbres qui vont s'entraîner sur des sous parties différentes de la base d'apprentissage, puis faire voter ces sous-modèles afin de prendre la décision finale en faisant appel au *boosting*.

L'idée du *boosting* c'est d'entraîner l'un après l'autre, plusieurs modèles relativement faibles, en demandant à chaque modèle d'essayer de corriger les erreurs effectuées par son prédécesseur, alors le modèle améliore sa capacité prédictive en apprenant de ses erreurs. Dans ce cas, on obtient un ensemble de modèles parfaitement complémentaires dans lequel les faiblesses des uns sont compensées par les forces des autres. Les modèles sont tous relativement faibles, ils sont chacun en situation de sous-apprentissage, mais en les construisant les uns par dessus les autres, on est capable de réduire le biais du groupe.

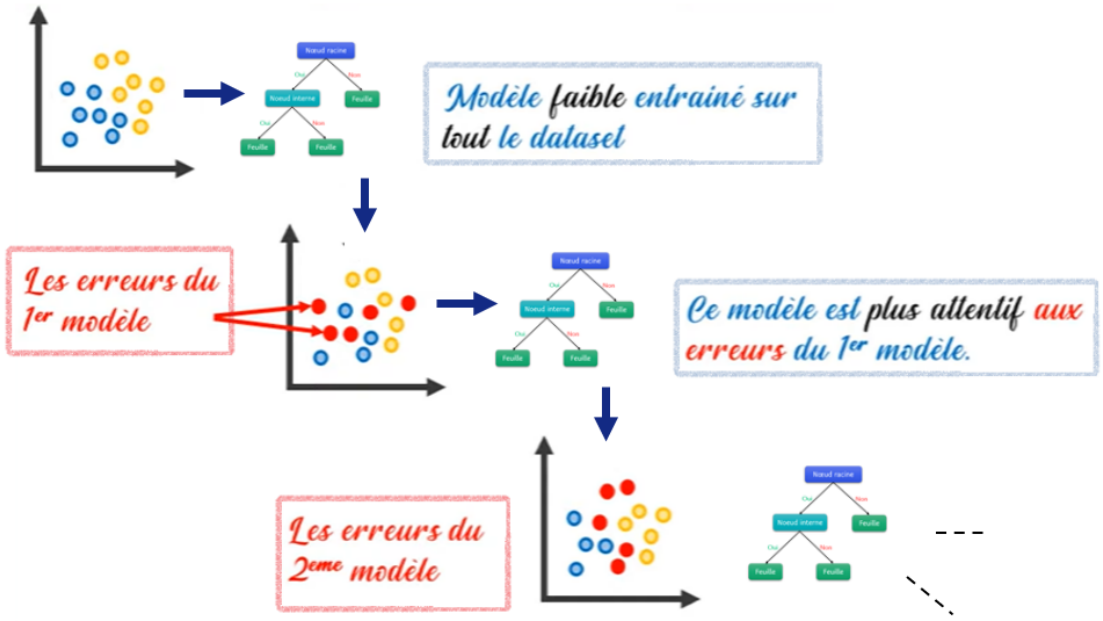

Figure 2.18 : Représentation du boosting dans le modèle XGBoost.

• **Optimisation de XGBoost**

Pour une meilleure performance, on utilisera la ***régression Tweedie***. La famille de distributions Tweedie est une sous-classe de la famille exponentielle (modèles de dispersion exponentielle) avec une relation spéciale moyenne-variance. Pour $X \rightsquigarrow Tw_p(\mu, \sigma^2)$ on a $E[X] = \mu$ et $Var(X) = \sigma^2.\mu^p$ avec $1 < p < 2$ le paramètre de puissance de Tweedie.

Afin d'optimiser le modèle de XGBoost par la méthode ***Random search*** (méthodes d'optimisation numérique semblable à Grid Search mais ne nécessitent pas d'optimiser le gradient), il faut bien choisir les hyper-paramètres du modèle Decision Tree et aussi les hyper-paramètres propres au Gradient Boosting :

◦ La puissance de la variance de tweedie (*tweedie_variance_power*).

◦ la proportion de variables explicatives utilisées à chaque nouvel arbre créé (*colsample_bytree*).

◦ la valeur minimale de réduction de la fonction d'erreur requise pour réaliser une nouvelle partition d'un noeud dans les arbres (*gamma*).

◦ La valeur minimale de la somme des poids contenue par un nouvel arbre créé (*min_child_weight*).

◦ Le taux d'apprentissage (*eta*).

2.3 Méthodes de Deep Learning en provisionnement

2.3.1 Réseaux de neurones (Neural networks)

Un réseau de neurones artificiels (Wikistat (2016)) est une application, non linéaire par rapport à son paramètres θ qui associe à une entrée x une sortie $y = f(x, \theta)$. Les réseaux de neurones peuvent être utilisés pour la régression ou la classification et les paramètres sont estimés à partir d'un échantillon d'apprentissage. La fonction à minimiser n'est pas convexe, ce qui conduit à des minimiseurs locaux. Le succès de la méthode provenait d'un théorème d'*approximation universelle* dû à Cybenko (1989) et Hornik (1991). De plus, Cun (1986) a proposé une méthode efficace de calculer le gradient d'un réseau de neurones, appelé *rétropropagation du gradient*, qui permet d'obtenir un minimiseur local du critère quadratique.

• **Neurone artificiel**

Le neurone artificiel est caractérisé par des valeurs d'entrées, des poids associés à chacune de ces valeurs, une fonction de combinaison des poids et de leur valeur, une fonction d'activation (appelée aussi, fonction de transfert) et enfin une valeur de sortie. On peut modéliser le neurone artificiel par une fonction f_j de l'entrée $x = (x_1, ..., x_d)$ pondérée par un vecteur de poids de connexion $w_j = (w_{j,1}, ..., w_{j,d})$, complété par un neurone bias b_j, et associé à une fonction d'activation ϕ à savoir

$$y_j = f_j(x) = \phi(< w_j, x > +b_j).$$

Plusieurs fonctions d'activation peuvent être envisagées :

▷ La fonction linéaire (identité) $\phi(x) = x$.

▷ La fonction softmax (exponentielle normalisée) $\phi_j(x) = \frac{\exp(x_j)}{\sum_{k=1}^{d} \exp(x_k)}$.

▷ La fonction sigmoïde $\phi(x) = \frac{1}{1+\exp(-x)}$.

▷ La fonction tangente hyperbolique $\phi(x) = \tanh(x) = \frac{\exp(2x)-1}{\exp(2x)+1}$.

▷ La fonction ReLU (Rectified Linear Unit) $\phi(x) = \max(0, x)$.

▷ La fonction ELU (Exponential Linear Unit) $\phi_\alpha(x) = x.1_{\{x\geq 0\}} + \alpha(\exp(x) - 1).1_{\{x<0\}}$.

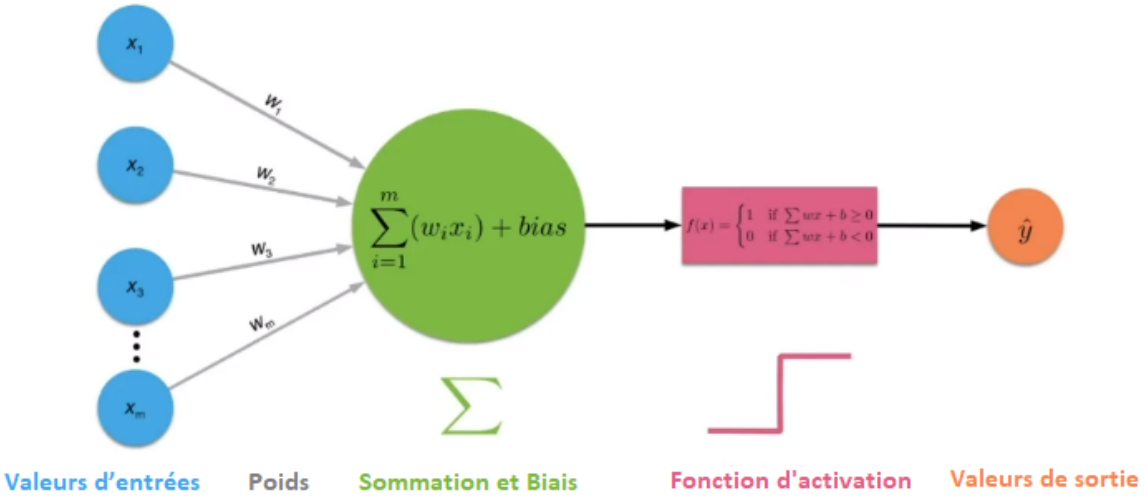

FIGURE 2.19 : Représentation schématique d'un neurone artificiel où $\Sigma =< w_j, x > +b_j$.

Le sigmoïde était la fonction d'activation la plus utilisée puisqu'il est différentiable et permet de garder les valeurs dans l'intervalle $[0, 1]$. Néanmoins, il est problématique car son gradient est très proche de 0 lorsque $|x|$ n'est pas proche de 0. Les réseaux de neurones avec un nombre élevé de couches (ce qui est le cas en Deep Learning), cause des problèmes à l'algorithme de *rétropropagation* (Backpropagation) pour estimer les paramètres.

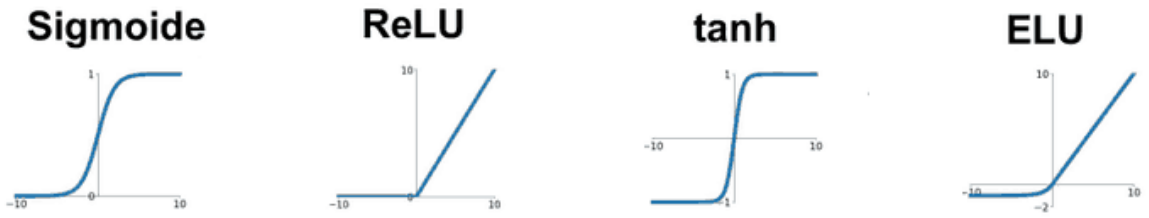

FIGURE 2.20 : Représentation de quelques fonctions activations.

C'est pourquoi la fonction sigmoïde a été remplacée par la fonction ReLU. Cette fonction n'est pas dérivable en 0 mais en pratique ce n'est pas vraiment un problème puisque la probabilité d'avoir une entrée égale à 0 est généralement nulle.

- **La rétro-propagation**

Le concept de rétro-propagation (Backpropagation) aide les réseaux de neurones à améliorer leur précision. Lorsque les réseaux de neurones sont formés, un ensemble de valeurs d'entrée est transmise avec la valeur de sortie attendue correspondante. Les fonctions d'activation produisent alors une sortie à partir de l'ensemble des entrées. Lorsque le résultat réel est différent du résultat attendu, les poids appliqués aux neurones sont mis à jour : les informations sont renvoyées à nouveau dans le réseau de neurones, alors les poids et biais sont améliorés.

• **Perceptron Multi-Couche (PMC)**

Un perceptron multicouche est une structure composée de plusieurs couches cachées de neurones dont la sortie d'un neurone d'une couche devient l'entrée d'un neurone de la couche suivante. En outre, la sortie d'un neurone peut être aussi l'entrée d'un neurone de la même couche ou d'un neurone des couches précédentes : c'est le cas pour les réseaux de neurones récurrents. Sur la dernière couche, appelée couche de sortie, nous pouvons appliquer une fonction d'activation différente de celle des couches cachées selon le type de problèmes que nous avons à résoudre : régression ou classification.

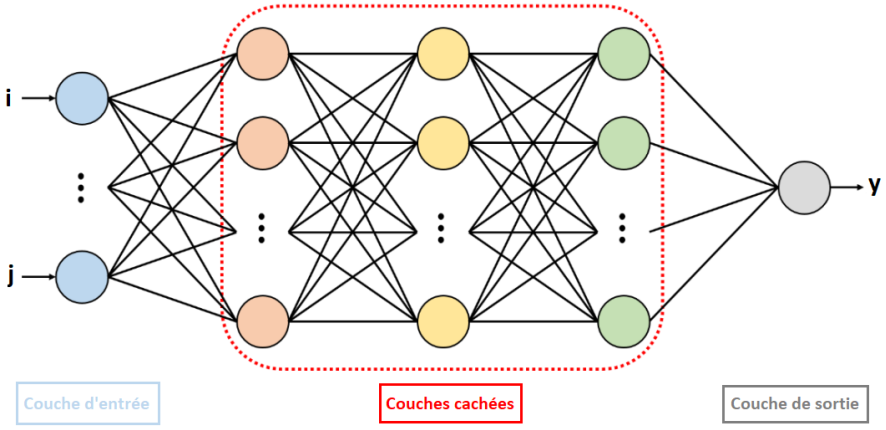

FIGURE 2.21 : Représentation d'un perceptron multi-couche.

Dans le cas de la régression, nous n'appliquons aucune fonction d'activation sur la couche de sortie.

Soit un perceptron multicouches à N couches cachées, on note $h^{(o)}(x) = x$ avec x la variable d'entrée. Pour $k \in [\![1, N]\!]$ (Couches cachées)

$$a^{(k)}(x) = b^{(k)} + W^{(k)} h^{(k-1)}(x),$$
$$h^{(k)}(x) = \phi\left(a^{(k)}(x)\right).$$

Pour $k = N + 1$ (Couche de sortie)

$$a^{(N+1)}(x) = b^{(N+1)} + W^{(N+1)} h^{(N)}(x),$$
$$h^{(N+1)}(x) = \psi\left(a^{(N+1)}(x)\right) = f(x, \theta),$$

où ϕ est la fonction d'activation et ψ est la fonction d'activation de la couche de sortie. À chaque étape, $W^{(k)}$ est une matrice dont le nombre de lignes correspond au nombre de neurones de la couche k et le nombre de colonnes au nombre de neurones de la couche $k - 1$.

• **Estimation des paramètres**

Une fois l'architecture du réseau choisie, les paramètres (les poids w_j et les biais b_j) doivent être estimés à partir d'un échantillon d'apprentissage. L'estimation est obtenue en minimisant une fonction de perte avec un algorithme de descente de gradient. Nous devons d'abord choisir la fonction de perte.

⋆ **Fonction de perte**

Il est classique d'estimer les paramètres en maximisant la vraisemblance. Cela correspond à la minimisation de la fonction de perte qui est l'opposé du logarithme de la vraisemblance. Soient θ le vecteur des paramètres à estimer et ℓ la fonction coût, nous considérons la fonction de perte suivante

$$L(\theta) = \mathbb{E}_{(X,Y)\sim P}[\ell(f(X,\theta),Y)].$$

Dans le cas d'un problème de régression, la maximisation de la vraisemblance est souvent équivalente à la minimisation de la perte quadratique

$$L(\theta) = \mathbb{E}_{(X,Y)\sim P}\left(\|Y - f(X,\theta)\|^2\right).$$

Afin d'estimer les paramètres θ, nous utilisons un échantillon d'apprentissage $(X_i, Y_i)_{1\leq i\leq n}$ et nous minimisons la perte empirique

$$\tilde{L}_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(f\left(X_i,\theta\right), Y_i\right).$$

Nous ajoutons éventuellement un terme de régularisation. Cela conduit à minimiser le risque empirique pénalisé

$$L_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(f\left(X_i,\theta\right), Y_i\right) + \lambda\Omega(\theta).$$

Nous pouvons considérer une pénalisation $\mathbb{L}^2$

$$\Omega(\theta) = \sum_k \sum_i \sum_j \left(W_{i,j}^{(k)}\right)^2 = \sum_k \left\|W^{(k)}\right\|_F^2 \quad ,$$

où $\|W\|_F$ désigne la norme de Frobenius de la matrice W. Notez que seuls les poids sont pénalisés, les biais ne sont pas pénalisés. Il est facile de calculer le gradient de $\Omega(\theta)$

$$\nabla_{W^{(k)}}\Omega(\theta) = 2W^{(k)}.$$

Pour une pénalisation $\mathbb{L}^1$

$$\Omega(\theta) = \sum_k \sum_i \sum_j \left|W_{i,j}^{(k)}\right|.$$

Dans le but de minimiser le critère $L_n(\theta)$, un algorithme de ***descente de gradient stochastique*** (stochastic gradient descent) est utilisé, qui se déroule comme suit :

(1) Initialisation de $\theta = \left(W^{(1)}, b^{(1)}, \ldots, W^{(L+1)}, b^{(L+1)}\right)$.

(2) Pour N itérations :

- Pour chaque donnée d'apprentissage (X_i, Y_i)

$$\theta = \theta - \varepsilon\frac{1}{m}\sum_{i\in B}\left[\nabla_\theta \ell\left(f\left(X_i,\theta\right), Y_i\right) + \lambda\nabla_\theta\Omega(\theta)\right].$$

Dans l'algorithme précédent, on ne calcule pas le gradient de la fonction de perte à chaque étape de l'algorithme mais uniquement sur un sous-ensemble B de cardinalité m appelée ***batch***. B est tiré aléatoirement sans remise. Une itération sur tous les données d'apprentissage est appelée ***epochs***. Le nombre total d'itérations est égal au nombre d'*epochs* multiplié par la taille de l'échantillon n divisé par m, appelée ***size of a batch***. Cette procédure est appelée ***Batch Learning***.

Le taux d'apprentissage ϵ (***learning rate***) est un paramètre important dans la descente de gradient : si la valeur de ϵ est faible, alors la convergence vers un extremum peut être très lente et nécessiter un nombre d'itérations très important. En revenche, si la valeur de ϵ est importante, alors l'algorithme peut dépasser un minimum global ou même à diverger.

⋆ Optimisation des réseaux de neurones

Il est nécessaire, avant l'entraînement du réseau de neurones, de fixer certains hyper-paramètres :

Le nombre de couches cachées : plus le nombre de couches cachées augmente, plus le modèle est meuilleur mais complexite. En revanche, avoir un grand nombre de couches cachées peut parfois conduire à des problèmes de gradient (*Vanishing gradient* & *Exploding gradient*).

Le nombre de neurones présents dans chacune des couches cachées : lorsque les neurones sont nombreux, la complexité augmente.

Ces paramètres peuvent prendre une infinité de valeurs, il est donc impossible de tester toutes les combinaisons possibles. On peut utiliser la méthode *Grid Search* pour automatiser la recherche en fixant un intervalle de valeurs en amont.

2.3.2 Mixture Density Network (MDN)

• Densité mélange (Mixture Density)

Une densité mélange (loi mélange) est une fonction de densité qui est issue d'une combinaison convexe de plusieurs fonctions de densité. La forme génerale d'une densité mélange s'écrit

$$f(x) = \sum_{k=1}^{n} \Pi_k f_k(x),$$

avec Π_k les proportions respectives des sous densités telle que $0 < \Pi_k \leqslant 1$ et $\sum_{k=1}^{n} \Pi_k = 1$ et f_k la densité du k^{ieme} composant.

Un modèle de mélange gaussien est un mélange de plusieurs composantes normales. Il peut approximer n'importe quelle distribution avec suffisamment de composants, pour une précision souhaitée.

• Réseau de densité (Density Network)

Un réseau de densité est un réseau de neurones artificiels dont l'objectif n'est pas simplement d'apprendre à produire une valeur continue unique, mais d'apprendre à produire les paramètres de distribution (par exemple : la moyenne et l'écart type) en fonction de certaines caractéristiques d'entrée. La prédiction de la distribution par rapport à une valeur unique présente des avantages, comme la possibilité de donner des limites d'incertitude à la prédiction. Il s'agit d'une approche ***bayésienne*** de la résolution d'un problème de régression.

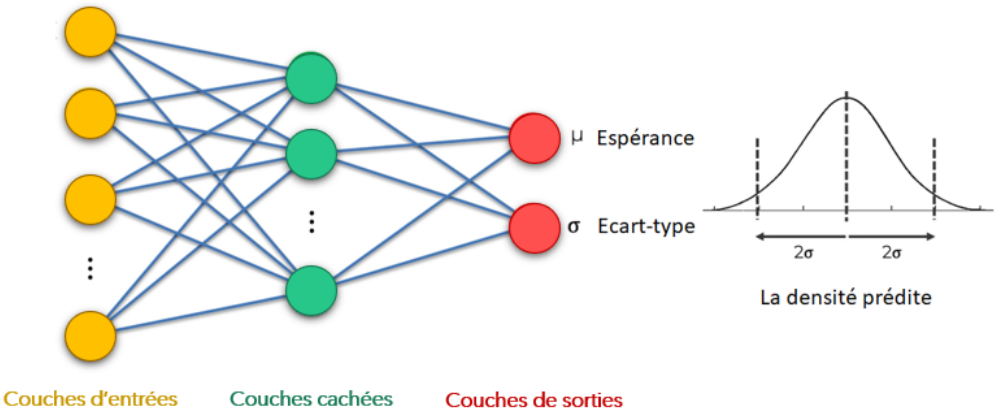

FIGURE 2.22 : Représentation d'un réseau de densité.

• **Réseau de densité mélange (Mixture Density Network)**

Le réseau de densité mélange ou *Mixture density network* (MDN) (Bishop (1994)) est un type de réseau de neurones artificiels, qui utilise l'hypothèse que toute distribution générale, peut être décomposée en un mélange de distributions normales $\mathcal{N}(\mu, \sigma^2)$. Le mélange peut également être personnalisé avec d'autres types de distributions. La densité de probabilité de la variable à prédire est alors représentée comme une combinaison linéaire de fonctions noyau de la forme

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_{i=1}^{K} \Pi_i(\boldsymbol{x}) \phi_i(\boldsymbol{y} \mid \boldsymbol{x}),$$

où K est le nombre de composants du mélange et $\Pi_i(\boldsymbol{x})$ est appelé coefficients de mélange, en utilisant les fonctions noyaux qui sont des gaussiennes de la forme

$$\phi_i(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{(2\pi)^{d/2} \sigma_i(\boldsymbol{x})^d} \exp\left\{ -\frac{\|\boldsymbol{y} - \boldsymbol{\mu}_i(\boldsymbol{x})\|^2}{2\sigma_i(\boldsymbol{x})^2} \right\},$$

où $\boldsymbol{\mu}_i$ représente l'espérance du i^{eme} noyau. En supposant que les composantes du vecteur de sortie sont statiquement indépendantes les unes des autres dans la distribution, et qu'elles peuvent être décrites par une variance commune $\sigma_i(\boldsymbol{x})$. L'hypothèse d'indépendance peut être relâchée en introduisant des matrices de covariance complètes pour chaque noyau gaussien. Cependant, un modèle de mélange gaussien avec ce noyau simplifié peut approcher n'importe quelle fonction de densité donnée avec une précision approximative, à condition que les coefficients de mélange et les paramètres gaussiens (moyennes et variances) soient correctement choisis.

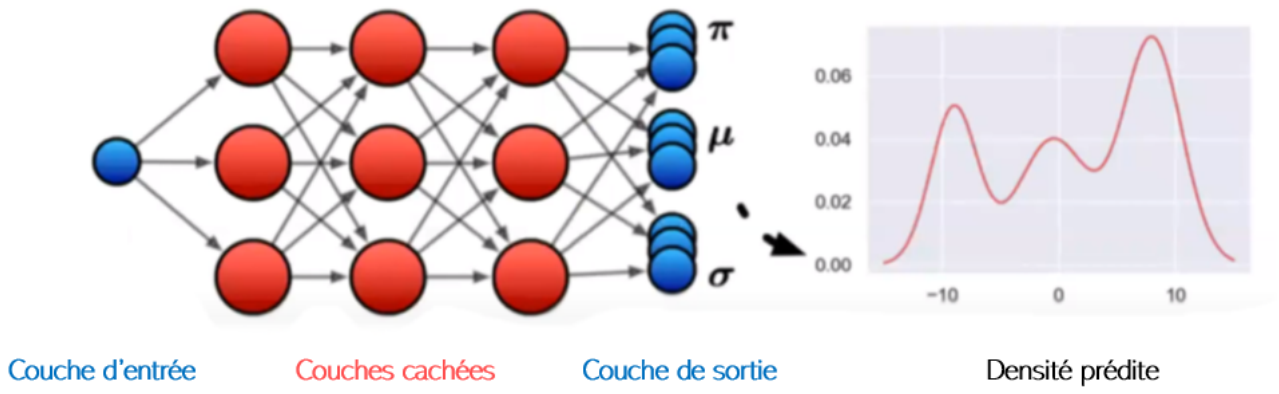

Figure 2.23 : Représentation d'un réseau de densité mélange.

La sortie du réseau détermine les paramètres d'un modèle de densité mélange. Par conséquent, le modèle de densité mélange représente la fonction de densité de probabilité conditionnelle de la variable prédite conditionnées par le vecteur d'entrée. Cette union entre le réseau neuronal traditionnel et la partie du modèle de mélange est réalisée en utilisant la log-vraisemblance de la combinaison linéaire des fonctions noyau comme fonction de perte du réseau de neurones.

⋆ **Les fonctions d'activation utilisées**

Il y a quelques restrictions pour différents paramètres à satisfaire :

1) Il est important que les coefficients de mélange Π_i satisfassent à la contrainte $\sum_{i=1}^{K} \Pi_i = 1$. Pour réaliser cette restriction, il suffit en principe d'avoir une fonction d'activation ***softmax*** dans les nœuds correspondant à Π_i.

2) Comme la variance σ_i représente des paramètres d'échelle, il est recommandé de les représenter en termes d'exponentielle de la sortie du réseau correspondant z_i^σ

$$\sigma_i = \exp\left(z_i^\sigma\right).$$

3) Les espérances μ_i représentent les paramètres de localisation. Si l'on tient compte de la notion d'antériorité non informative, on peut penser que ces paramètres sont directement représentés par les sorties du réseau, à savoir

$$\mu_{i,k} = z_{i,k}^\mu.$$

ce qui correspondrait, dans un cadre bayésien, au choix d'un a priori bayésien non informatif, en supposant que les sorties du réseau correspondant z_i^σ ont une distribution de probabilité uniforme, l'utilisation de cette représentation permet d'éviter les configurations pathologiques dans lesquelles une ou plusieurs des variances deviennent nulles.

Bishop (1994) a suggéré d'utiliser une fonction d'activation ***exponentielle*** pour le paramètre de variance. Ce choix a ses avantages. La fonction exponentielle tend vers une sortie positive et, du côté inférieur, elle n'atteint jamais vraiment zéro. Mais en pratique, elle présente quelques problèmes. La fonction exponentielle devient très grande très rapidement et dans le cas de jeux de données avec une variance élevée, l'apprentissage devient instable.

Guillaumes (2017) propose une autre alternative, une version modifiée d'une activation ***ELU***. Cette fonction conserve le comportement exponentiel et revient à un comportement linéaire pour les valeurs supérieures. Le seul problème est que le comportement exponentiel se produit lorsque l'entrée x est négatif. Mais si on ajoute 1 à cette fonction, on obtient une fonction qui s'approche du comportement exponentiel. Axel Brando Guillaumes a donc proposé d'utiliser la fonction $\boldsymbol{ELU(.)+1}$ comme fonction d'activation pour le paramètre de variance. Puisque ce paramètre peut aussi devenir nul, nous avons ajouté un epsilon à l'ELU modifiée pour en assurer la stabilité. Ainsi, l'activation finale utilisée est : $\boldsymbol{ELU(.)+1+\epsilon}$ avec $\boldsymbol{\epsilon = 10^{-15}}$.

⋆ La fonction de perte (Loss Function)

Le réseau est entraîné en utilisant la rétro-propagation standard. Pour définir une fonction d'erreur, à utiliser comme fonction de perte, l'approche standard est la méthode du maximum de vraisemblance, qui exige la maximisation de la fonction de log-vraisemblance ou, de manière équivalente, la minimisation du logarithme négatif de la vraisemblance. Par conséquent, la fonction de perte pour le réseau de densité de mélange est la suivante

$$\log \mathcal{L}(\boldsymbol{y} \mid \boldsymbol{x}) = -\log(p(\boldsymbol{y} \mid \boldsymbol{x})) = -\log\left(\sum_{i=0}^{K} \Pi_i(\boldsymbol{x})\phi_i(\boldsymbol{y} \mid \boldsymbol{x})\right),$$

où $\phi_i(\boldsymbol{y} \mid \boldsymbol{x})$ est identique à la fonction du noyau gaussien. Le terme $\sum p(\boldsymbol{x})$ a été abandonné car il est indépendant des paramètres du modèle de mélange, et donc indépendant des poids du réseau. Ainsi, l'objectif des réseaux à densité mélange est de modéliser la densité de probabilité conditionnelle complète des variables de sortie.

⋆ Minimiser la fonction de perte

Une fois l'architecture de notre réseau de neurones déterminée, il faut trouver un moyen pour minimiser la fonction de perte et modifier les poids afin d'obtenir un résultat optimal. Pour ce faire, nous devons calculer les dérivées de la fonction de perte par rapport aux poids du réseau neuronal. Une méthode permettant de résoudre ce problème consiste à utiliser la procédure standard de rétropropagation, à condition d'obtenir des expressions appropriées pour les dérivées de l'erreur par rapport aux activations des unités de sortie du réseau de neurones.

Ces dernières années, de nombreux nouveaux algorithmes d'optimisation par descente de gradient ont été développés, tels que *Nesterov accelerated gradient*, *Adagrad*, *Adadelta*, *RMSprop* et *Adam*.

⋆ **Optimisation du MDN dans le cadre de provisionnement**

La conception du MDN utilise la métrique *MSE* pour optimiser directement les sinistres payés dans la couche de sortie. Le MDN surpasse généralement les réseaux de neurones classiques, car une prévision moyenne peut être extraite de la distribution mélange (l'esperence de la densité prédite). En outre, l'hypothèse d'hétéroscédasticité (variance non constante de la réponse) aide à produire des prévisions plus stables. Un inconvénient des MDN est la grande difficulté de convergence, en 2020 Kuo (2020) confirme la difficulté de la convergence d'une distribution mélange log-gaussienne. En outre, bien que les gaussiennes mélange puissent approximer n'importe quelle distribution, le modèle est techniquement une paramétrisation de la distribution de sortie et sera donc limité dans sa modélisation. L'utilisation d'un plus grand nombre de gaussiennes dans le mélange peut pallier ce problème, mais au prix d'une énorme complexité et temps de calcul.

Dans le cadre de provisionnement appliqué sur les données aggrégées, les variables d'entrée du MDN sont i et j, les origines d'accident et de développement, respectivement et seront normalisées pour une bonne performance. Les hyperparamètres suivants seront testés pour trouver l'architecture la plus performante :

▷ **Régularisation de poids λ_w**

La régularisation $L2$ est généralement imposée aux poids du réseau pour réduire la flexibilité du modèle, ce qui réduit le sur-apprentissage. Des poids importants peuvent provoquer de grandes fluctuations dans la sortie sous des entrées légèrement différentes, ce qui rendra les projections instables. Le fait de disposer d'un petit ensemble de données encourage également le sur-apprentissage, puisqu'il est possible d'ajuster un plus grand nombre de fonctions avec une erreur de test faussement faible, alors qu'en réalité les projections sont inexactes sur des données non vues. Le triangle de développement superieur est toujours considéré comme un petit ensemble de données dans le cadre du réseau de neurones, ce qui rend encore plus nécessaire la régularisation. Les ensembles de données présentant une tendance plus simple nécessiteront généralement une régularisation plus importante afin de pénaliser la complexité inutile. Les coefficents de pénalité $L2$ ont été testés sur une Les coecients de pénalité L2 ont été testés sur une échelle logarithmique échelle logarithmique afin de tester efficacement un large espace de paramètres : $\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$.

▷ **Régularisation de l'activité (λ_σ)**

Un optimum local dans la fonction de perte peut impliquer une variance déraisonnablement élevée, ce qui produit des résultats irréalistes. La régularisation de l'activité pénalise la sortie σ, plutôt que les poids précédant le nœud de sortie. Les ensembles de données présentant une tendance plus complexe nécessiteront généralement une régularisation d'activité à sigma plus élevé, afin d'encourager le MDN à capturer la tendance. Les valeurs testées sont $\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$.
Soit w un vecteur de tous les poids du réseau, à l'exclusion des poids de biais. Soit λ_w et λ_σ les coefficients de pénalité L2 de poids et de sigma, respectivement. Avec les pénalités appliquées ci-dessus, la fonction de perte pendant l'apprentissage suit l'équation suivante

$$\begin{aligned}\text{Loss}\left(\mathbf{X}_{\text{Apprentissage}}, \hat{\mathbf{X}}_{\text{Apprentissage}} \mid \mathrm{w}, \lambda_w, \lambda_\sigma\right) = &-\frac{1}{|\mathbf{X}_{\text{Apprentissage}}|} \sum_{i,j:X_{i,j}\in\text{Apprentissage}} \ln\left(f_{\hat{X}_{i,j}}\left(X_{i,j} \mid \mathrm{w}\right)\right) \\ &+ \lambda_w||\mathrm{w}||^2 + \lambda_\sigma \sum_{i,j:X_{i,j}\in\text{Apprentissage}} \sum_{k=1}^{K} \sigma_{i,j,k}^2,\end{aligned}$$

avec X la charge incrémentale réelle et $\hat{X}$ la charge incrémentale prédite.

▷ **Dropout (p)**

Le dropout est une technique de régularisation couramment utilisée dans les réseaux neuronaux pour minimiser le sur-apprentissage. Le dropout a souvent été appliqué avec succès. dans les modèles de Gabrielli (2019) et Kuo (2019). Soit un dropout appliqué à une couche cachée spécifique, pendant l'apprentissage, à chaque époque, le MDN fixera la sortie des nœuds individuels de cette couche à 0, avec un taux p. Cela empêche le réseau de sur-apprendre les neurones en leur faisant subir un apprentissage excessive. Cela empêche le réseau de sur-former les neurones en leur attribuant trop de poids par rapport au reste du réseau. Dans ce cas, le réseau sera forcé d'accorder de l'importance à d'autres neurones. Ceci réduit le sur-apprentissage et augmente la stabilité de la sortie du MDN. Lorsque le MDN a été entraîné et qu'il produit des projections, le dropout n'est pas appliqué. Cependant, la sortie de chaque neurone sera multipliée par $1-p$, ce qui donne un effet similaire à l'assemblage de modèles. En dépit de ce cadre théorique, les premières modélisations ont montré que l'absence de dropout ou un faible dropout donnait généralement de meilleurs résultats. de faibles dropout, étaient généralement plus performants. Les valeurs de dropout testés étaient $[0, 0.1, 0.2]$. La capacité de modélisation du MDN étant réduite, le nombre de neurones a toujours été augmenté de $\frac{1}{1-p}$ lorsqu'un dropout de p est appliqué au réseau.

▷ **Nombre de composants de la densité (K)**

L'augmentation du nombre de composants améliorera la capacité de la densité mélange à se rapprocher de la vraie distribution, au prix d'un risque plus important de sur-apprentissage et de sur-paramétrisation. Les distributions unimodales sont couramment utilisées pour modéliser les provisions, il faut donc moins de composantes pour obtenir un ajustement satisfaisant. Une gamme de 1 à 4 composants s'est avérée adéquate pour l'approximation de la distribution.

▷ **Nombre de couches cachées (h)**

Entre 1 à 4 couches cachées ont été testées et se sont avérées suffisantes pour obtenir un ajustement satisfaisant. Cette fourchette est conforme à la littérature actuarielle (Wüthrich (2018), Gabrielli (2020b), Cybenko (2020)) et à la littérature appliquant les MDN (Bishop (1994), Zen (2014), Ormoneit et Tresp (1996)). Selon la théorie de l'approximation universelle, expliquée par Cybenko (1989), une couche cachée est suffisante pour approximer toute fonction continue. Malgré cette théorie, il peut falloir de nombreux nœuds pour modéliser une fonction précise, d'où l'utilisation de plus d'une couche pour capturer plus efficacement les interactions profondes entre les variables. Un nombre excessif de couches cachées peut conduire au *Vanishing Gradient Problem* (divergence du gradient), ce qui ralentit le processus de modélisation.

▷ **Nombre de neurones (n)**

De même, pour les couches cachées, un faible nombre de neurones réduit la flexibilité du modèle, ce qui peut entraîner un ajustement moins précis. Un nombre trop élevé de neurones peut entraîner un sur-apprentissage. Une fourchette de 10 à 30 neurones pour chaque couche était compatible avec la littérature actuarielle pour le provisionnement non-vie. Cependant, cette littérature s'est concentrée sur la modélisation de l'estimation centrale (BE), dont les résultats sont plus simples. Dans ce mémoire, une gamme de 20 à 100 neurones a donné les résultats les plus précis. Pour simplifier le processus de sélection des hyperparamètres, toutes les couches cachées ont été configurées pour contenir le même nombre de neurones.

▷ **Fonctions d'activation des couches cachées**

Les fonctions d'activation doivent être non linéaires, car une fonction d'activation linéaire est équivalente à la sommation pondérée qui se produit déjà entre chaque couche. De plus, ces fonctions permettront de capturer les non-linéarités entre les variables. Les fonctions *tanh* et *sigmoïde* sont simples, courantes et faciles à mettre en œuvre. Ces deux fonctions peuvent souffrir de *Vanishing Gradient Problem* avec des réseaux plus profonds, ce qui ralentit l'apprentissage. La fonction d'activation ReLU est également courante et a été démontrée par Kuo (2019) ainsi que par Rossouw et Richman (2019). Pour simplifier le processus de sélection de l'architecture du modèle, les premiers tests ont montré que la fonction d'activation sigmoïde était la plus performante, et c'est donc cette fonction d'activation a été utilisée pour toutes les modélisations ultérieures.

Une limite de 10000 époques a été fixée lors de l'exécution de l'algorithme d'optimisation des hyperparamètres, afin d'augmenter l'efficacité.

Chapitre 3

Présentation de données utilisées

Ces dernières années, l'application du Machine Learning et du Deep Learning au provisionnement en d'assurance non-vie a connu une croissance rapide. Ces méthodes sont particulièrement utiles lorsqu'elles sont appliquées à de grands jeux de données, tels que des sinistres individuels ou des triangles de développement de sinistres importants. Malheureusement, ces grandes jeux de données sont rares dans la littérature actuarielle sous prétexte de confidentialité des données des entreprises ou des clients. Il faut donc se tourner vers les données simulées. Bien que l'objectif ultime de ces méthodes soit l'application aux données réelles, l'utilisation de données simulées contenant des caractéristiques couramment observées dans les données réelles est également à privilégier. Dans ce chapitre, on va présenter deux types de données qui seront utilisées dans la partie application : les données réelles de ADDACTIS France et les données simulées par SynthETIC sous R, quatre environnements ont été simulés avec des caractéristiques et des complexités différentes.

3.1 Données simulées par SynthETIC (R)

Bien qu'il existe un certain nombre de simulateurs de sinistres (Data Maker, ...), chacun étant utile dans son propre contexte, l'inclusion d'un certain nombre de caractéristiques de données souhaitables (mais compliquées) nécessite un développement supplémentaire. C'est pourquoi, dans ce mémoire, nous avons utilisé le nouveau simulateur de sinistres individuels et de triangles de développement en assurance non-vie appelé "**SynthETIC**" (2020), qui est disponible gratuitement (open source) sur R.

SynthETIC est développé par Avanzi et Taylor (2020) . Il est considéré comme étant un simulateur de sinistres individuels (*Charges i.e paiement + D/D*) qui génère diverses caractéristiques des sinistres en assurance non-vie. Un jeu initial de paramètres de test, conçu pour refléter l'expérience d'un portefeuille de ***responsabilité civile automobile*** (RC auto), a été mis en place et appliqué par défaut pour générer un jeu de données de test réaliste de sinistres. Le jeu de données simulées permet ensuite de tester a posteriori la validité de divers modèles de provisionnement et de valider et/ou rejeter certaines hypothèses actuarielles formulées dans la modélisation des sinistres. Les hypothèses de distribution utilisées pour générer cet ensemble de données peuvent être facilement modifiées par les utilisateurs pour correspondre à leurs expériences.

La dimension de chaque triangle de développement simulé de charges incrémentées est de **40x40** : 40 trimestres d'observations pour 40 trimestres de développements. Ce choix permet de tester les prédictions des modèles à court, au moyen et à long terme. La simulation de différents environnements va permettre de tester la capacité de des modèles à détecter les tendances complexes et à produire des prévisions dans divers environnements difficiles.

- **Paramètres**

◦ N_i **:** Le nombre de sinistres au cours du trimestre d'accident i.

◦ $S_{i,r}$ **:** Charge total du sinistre r survenu au cours du trimestre d'accident i.

◦ μ_R **:** Délai moyen de déclaration de chaque sinistre.

◦ σ_R **:** Ecart-type du délai de déclaration de chaque sinistre.

◦ μ_S **:** Délai moyen de traitement de chaque sinistre.

◦ σ_S **:** Ecart-type du délai de traitement de chaque sinistre.

◦ SI_i^{OCC} **:** L'inflation trimestrielle survenus au cours du trimestre d'accident i.

◦ SI_c^{PAY} **:** L'inflation trimestrielle superposée pour le trimestre calendaire c.

◦ I **:** Nombre de périodes de développement des sinistres considérées, qui est égal au nombre d'années (*10 ans*) divisé par l'unité de temps ($\frac{1}{4}$ i.e par trimestre).

◦ E **:** L'Exposition au risque associée à chaque période i.

◦ λ **:** Fréquence des sinistres par unité d'exposition pour la période i.

Les délais de déclaration et de traitement de sinistres suivent une distribution de Weibull. Le nombre de sinistres pour la période de l'accident est distribué selon la loi $\mathcal{P}(E \times \lambda)$.

3.1.1 Environnement 1 : branche de développement court

Cet ensemble de données comprend des sinistres simples, à queue très courte, dont la composition est homogène pour tous les trimestres d'accident. L'inflation trimestrielle de base est fixée à 2%. Les paramètres utilisés pour cet environnement sont les suivants :

▷ $N_i \sim \mathcal{P}(120000)$.

▷ $S_{i,r}^{0.2} \sim \mathcal{N}(9.5, 3)$.

▷ $\mu_R = 0.493$.

▷ $\frac{\sigma_R}{\mu_R} = 1.92$.

▷ $\mu_S = 4.12$.

▷ $\frac{\sigma_S}{\mu_S} = 1.23$.

▷ $SI_i^{OCC} = 1.02$.

▷ $SI_c^{PAY} = 1$.

Les charges de sinistres observées décroissent rapidement et se stabilisent à partir du 10ème trimestre de développement. Les ensembles de données de cette nature sont fréquents, par exemple la garantie automobile (dommages).

Une autre caractéristique de ce jeu de données réside dans le fait que le taux d'inflation trimestrielle de 2% reste constant. Les charges de sinistres incrémentaux ayant un faible bruit et une homogénéité presque parfaite, les méthodes usuelles prédisent l'estimation centrale presque parfaitement dans cet ensemble de données.

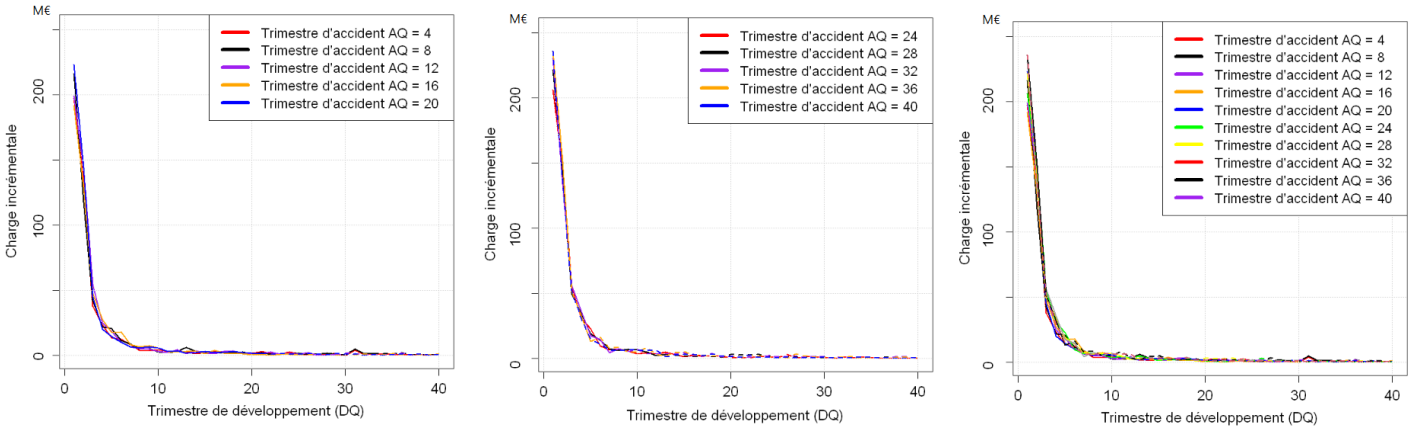

FIGURE 3.1 : Représentation du 4ème trimestre d'accident de chaque année de l'environnement 1.

Les lignes pleines dans le graphe ci-dessus représentent les données du triangle supérieur, tandis que les lignes pointillées représentent les données du triangle inférieur. La figure suivante représente la surface des charges incrémentales de l'environnement 1 en fonction des trimestres d'accident (AQ) et des trimestres de développement (DQ) :

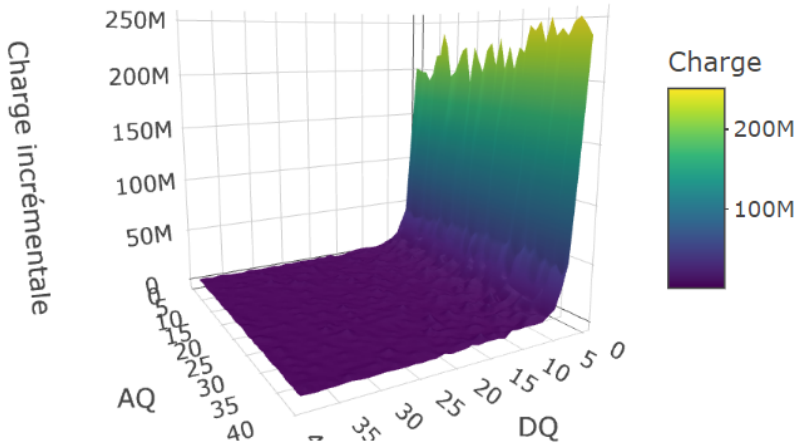

FIGURE 3.2 : Surface des données incrémentées de l'environnement 1.

3.1.2 Environnement 2 : variation de la cadence des sinistres

Cet ensemble de données introduit des complexités que *les méthodes usuelles ne parviennent pas à saisir*. On simule un passage progressif d'une branche de développement long à une branche de développement cours. Au début, il y a plus de sinistres à long terme, mais la proportion de ces sinistres diminue, tandis que la proportion de sinistres à court terme augmente. Deux jeux de données distincts sont simulés et ajoutés pour constituer le triangle de développement final, l'un avec les sinistres de longue durée et l'autre avec les sinistres de courte durée. L'inflation trimestrielle de base est fixée à 2%. L'inflation trimestrielle superposée est à taux contant de 3%. Les sinistres à court et à long terme sont désignés respectivement par X^{SHORT} et X^{LONG}.

Pour simuler les sinistres dans $\mathbf{X}^{SHORT}$, on utilise les paramètres suivants :

- $N_i^{SHORT} \sim \mathcal{P}\left(5000 + 55000\frac{i-1}{39}\right)$.
- $S_{i,r}^{0.25} \sim \mathcal{N}(9.5, 3)$.
- $\mu_R^{SHORT} = 0.493$.
- $\frac{\sigma_R^{SHORT}}{\mu_R^{SHORT}} = 1.92$.
- $\mu_S^{SHORT} = 6.58$.
- $\frac{\sigma_S^{SHORT}}{\mu_S^{SHORT}} = 1.02$.
- $SI_i^{OCC} = 1.02$.
- $SI_c^{PAY} = 1.03$.

Pour simuler les sinistres dans in $\mathbf{X}^{LONG}$, on utilise les paramètres suivants :

- $N_i^{LONG} \sim \mathcal{P}\left(60000 - 55000\frac{i-1}{39}\right)$.
- $S_{i,r}^{0.25} \sim \mathcal{N}(9.5, 3)$.
- $\mu_R^{LONG} = 2.47$.
- $\frac{\sigma_R^{LONG}}{\mu_R^{LONG}} = 1.54$.
- $\mu_S^{LONG} = 11.74$.
- $\frac{\sigma_S^{LONG}}{\mu_S^{LONG}} = 0.61$.
- $SI_i^{OCC} = 1.02$.
- $SI_c^{PAY} = 1.03$.

Ensuite, les deux jeux de données sont cumulés : $\mathbf{X} = \mathbf{X}^{SHORT} + \mathbf{X}^{LONG}$.

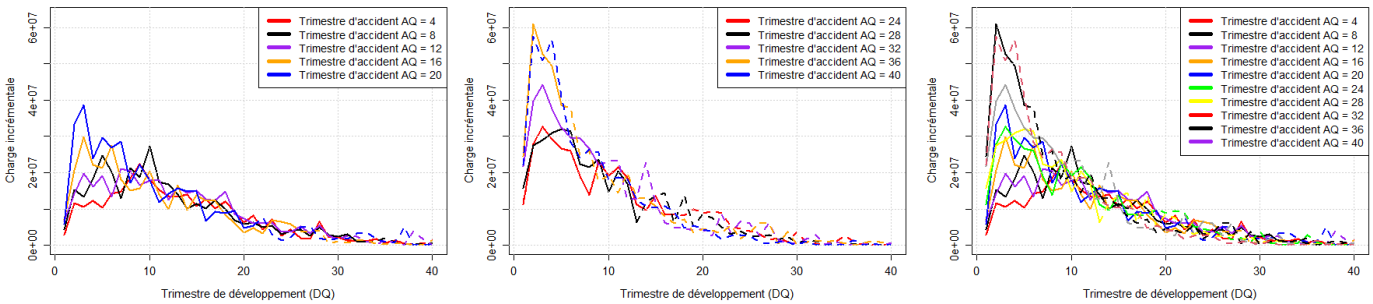

FIGURE 3.3 : Représentation du 4ème trimestre d'accident de chaque année dans l'environnement 2.

Dans ce jeu de données, les caractéristiques des sinistres changent progressivement de nature. Cette volatilité systématique se traduit de deux manières opposées. Premièrement, jusqu'au 10ème trimestre de développement (DQ), les sinistres augmentent en fonction des trimestres d'observation (AQ). Deuxièment, à partir du 10ème trimestre de développement, les sinistres diminuent en fonction des trimestres d'observation (AQ). Le fait de réduire la durée de traitement des sinistres explique ces deux tendances.

Les lignes pleines dans le graphe ci-dessus représentent les données du triangle supérieur, tandis que les lignes pointillées représentent les données du triangle inférieur. La figure suivante représente la surface de charges incrémentales de l'environnement 2, en fonction des trimestres d'accident (AQ) et des trimestres de développement (DQ) :

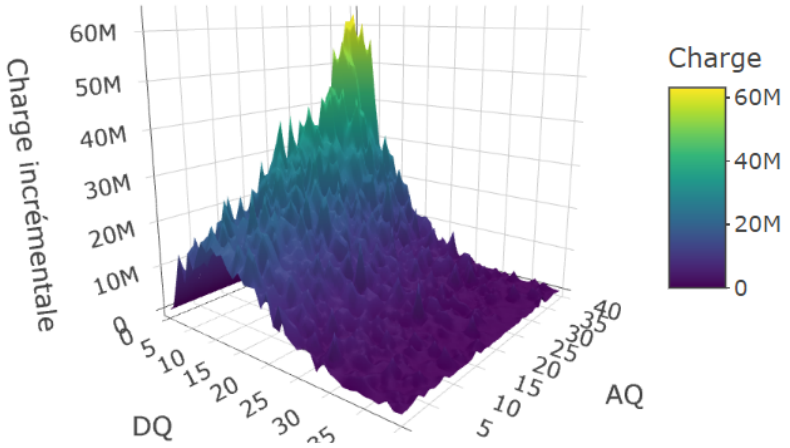

FIGURE 3.4 : Surface des données incrémentées de l'environnement 2.

3.1.3 Environnement 3 : branche de développement long avec choc d'inflation

Dans ce jeu de données, l'inflation trimestrielle superposée passe instantanément de 0% à 10%, après le 30ème trimestre. L'inflation trimestrielle de base est fixée à 2%.

Les paramètres utilisés pour cet environnement sont les suivants :

- $N_i \sim \mathcal{P}(6000)$.
- $S_{i,r}^{0.25} \sim \mathcal{N}(9.5, 3)$.
- $\mu_R = 2.47$.
- $\frac{\sigma_R}{\mu_R} = 1.54$.
- $\mu_S = 11.74$.
- $\frac{\sigma_S}{\mu_S} = 0.61$.
- $SI_i^{OCC} = 1.02$.
- $SI_c^{PAY} = 1_{\{c<30\}} + 1.1 \times 1_{\{c\geq 30\}}$.

Ce jeu de données teste la capacité des modèles à identifier les changements dans les effets du calendrier et à adapter les projections en conséquence. Seuls les 10 derniers trimestres calendaires dans le triangle supérieur contiennent des informations concernant le choc d'inflation, *ce qui augmente la difficulté de prédiction des les modèles.*

En outre, la méthode de ROCV est moins exposée aux données des derniers trimestres calendaires, en particulier dans la première partition. Ce jeu de données permet donc de tester si la méthode ROCV peut toujours être utilisée pour les derniers trimestres et capturer l'inflation.

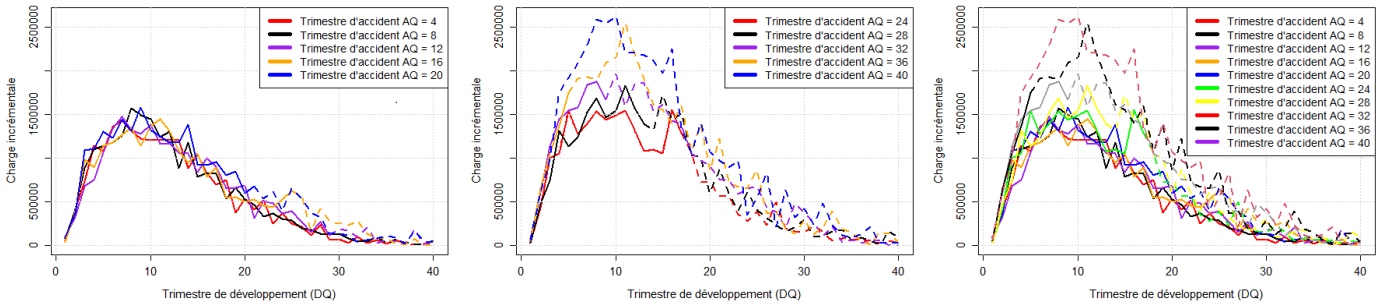

FIGURE 3.5 : Représentation du 4ème trimestre d'accident de chaque année dans l'environnement 3.

Les lignes pleines dans le graphe ci-dessus représentent les données du triangle supérieur, tandis que les lignes pointillées représentent les données du triangle inférieur. La figure suivante représente la surface de charges incrémentales de l'environnement 3, en fonction des trimestres d'accident (AQ) et des trimestres de développement (DQ) :

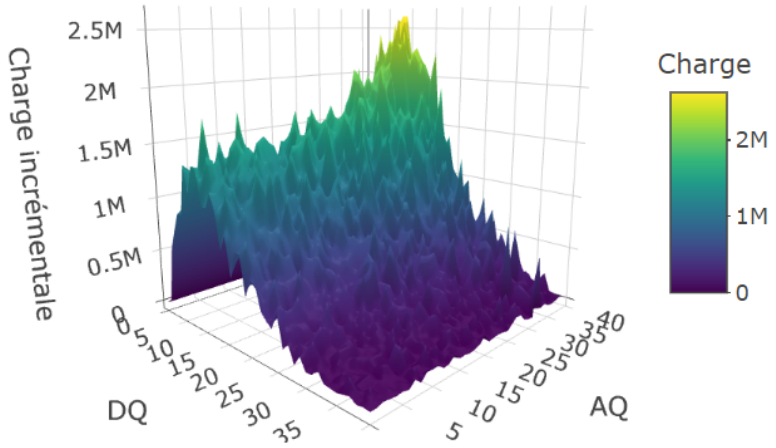

FIGURE 3.6 : Surface des données incrémentées de l'environnement 3.

3.1.4 Environnement 4 : branche de développement long à forte volatilité

Ce jeu de données représente le triangle de developpement par défaut généré par le simulateur SynthETIC. La volatilité des sinistres est considérablement élevée. Les sinistres sont très peu fréquents, mais suivent une distribution normale avec une puissance (*Power Normal Distribution*) de 0.2, ce qui se traduit par une sévérité très volatile. Plus la charge du sinistre augmente, plus le risque est élevé. Le délai de déclaration diminue, tandis que le délai de traitement augmente en moyenne.

Les petits sinistres sont traités légèrement plus rapidement car ils surviennent plus tard, mais cela s'arrête au 21ème trimestre. L'inflation trimestrielle superposée est de 30%, mais ce taux est multiplié par un facteur qui diminue à mesure que le montant de sinistre augmente. Ces tendances supplémentaires renforcent la volatilité qui était déjà présente. Il est normal que les sinistres d'un trimestre d'accident donnés suivent un schéma complètement différent (déclaration, traitement, volume, développement) de celui des sinistres trimestre d'accident adjacent. Une tendance prédomine dans ce jeu de données : l'évolution lente des sinistres et la forte inflation superposée.

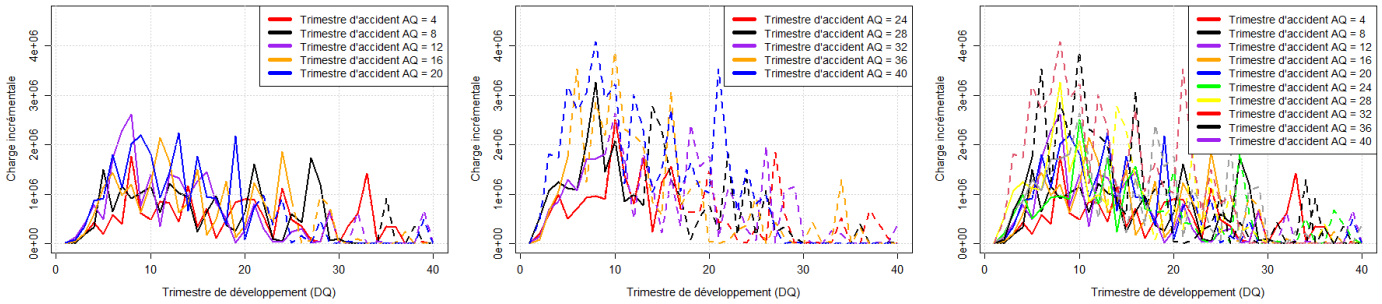

FIGURE 3.7 : Représentation du 4ème trimestre d'accident de chaque année dans l'environnement 4.

Les lignes pleines dans le graphe ci-dessus représentent les données du triangle supérieur, tandis que les lignes pointillées représentent les données du triangle inférieur. La figure suivante représente la surface de charges incrémentales de l'environnement 4, en fonction des trimestres d'accident (AQ) et des trimestres de développement (DQ) :

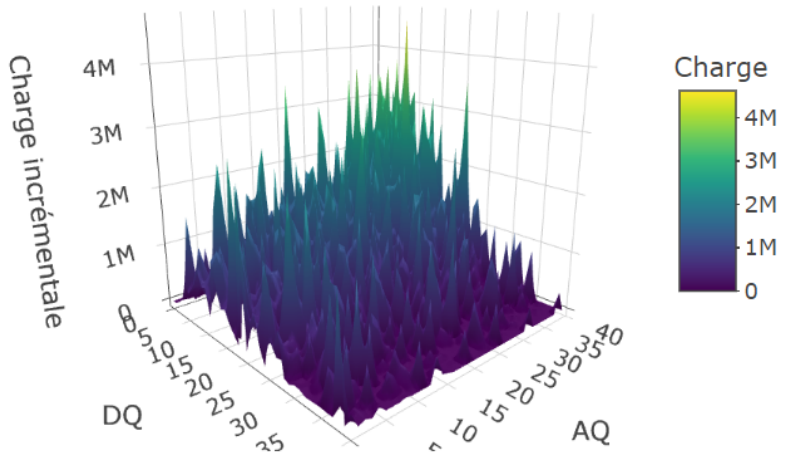

FIGURE 3.8 : Surface des données incrémentées de l'environnement 4.

3.2 Données réelles de ADDACTIS France

Les données réelle utilisées dans ce mémoire appartiennent à différentes branches d'assurance non-vie. La classification des différentes branches de l'assurance figure dans la partie du ***Code des Assurances*** rassemblant l'ensemble des décrets plus précisément à l'article ***R321-1***. Ces branches sont identifiées par des numéros allant de 1 jusqu'à 26, pour lesquelles on a l'agrément, c'est-à-dire l'autorisation d'exercer l'activité d'assurance. Pour les catégories d'assurance et états à produire (***Articles A344-2 à A344-4***), les opérations effectuées par les entreprises soumises au contrôle de l'Etat en vertu de l'article ***L. 310-1*** ou du 1° du III de l'article ***L. 310-1-1*** et par les fonds de retraite professionnelle supplémentaire mentionnés à l'article ***L. 381-1*** sont réparties en *39 catégories d'opérations*.

Les catégories utilisées pour la partie application sont :

22 ***Automobile (RC matériel)***.

23 ***Automobile (dommages)***.

28 ***Responsabilité civile générale***.

L'assurance automobile couvre les dommages causés "*avec*" ou "*à*" un véhicule automobile, les garanties habituelles sont la responsabilité civile du conducteur, les dommages corporels, l'incendie, le vol, les catastrophes naturelles, le bris de glace, etc. Aujourd'hui, cette branche est fortement exposée à des évolutions législatives, notamment sur l'indemnisation corporelle.

L'assurance responsabilité civile générale est indispensable pour chaque particulier ou entreprise pour se prémunir contre d'éventuels dommages corporels, matériels ou immatériels pouvant être causés à des tiers ou des clients lors de la réalisation de tâches liées à leur activité.

Les données réelles sont sous forme des triangles de règlements incrémentals, la dimension de chaque triangle est de **17x17** : 17 années d'observations (de 2004 à 2020) pour 17 années de développements.

Ces données viennent du cadre de modélisation de risque de souscription d'un assureur. L'objectif est d'appliquer les différents méthodes de ML et DL sur ces données, puis comparer les résultats obtenus avec ceux réalisés par les experts de l'équipe du modèle concerné. Les résultats dites experts utilisent les informations complètes du triangle, avec la possibilité de choix de coefficients et l'utilisation des facteurs de queue. Le modèle utilisé pour calibrer la distribution à l'ultime est le bootstrap Mack.

3.2.1 Automobile (RC matériel)

La garantie de responsabilité civile (RC) est obligatoire (article L211-1 du code des assurances). Le montant de la garantie est illimité pour le dommage corporel et plafonné à 100M€ pour le matériel. Cette garantie permet de compenser financièrement les dommages matériels subis par les tiers lorsque la responsabilité de l'assuré est engagée à la suite d'accident ou dégats causés par le véhicule assuré.

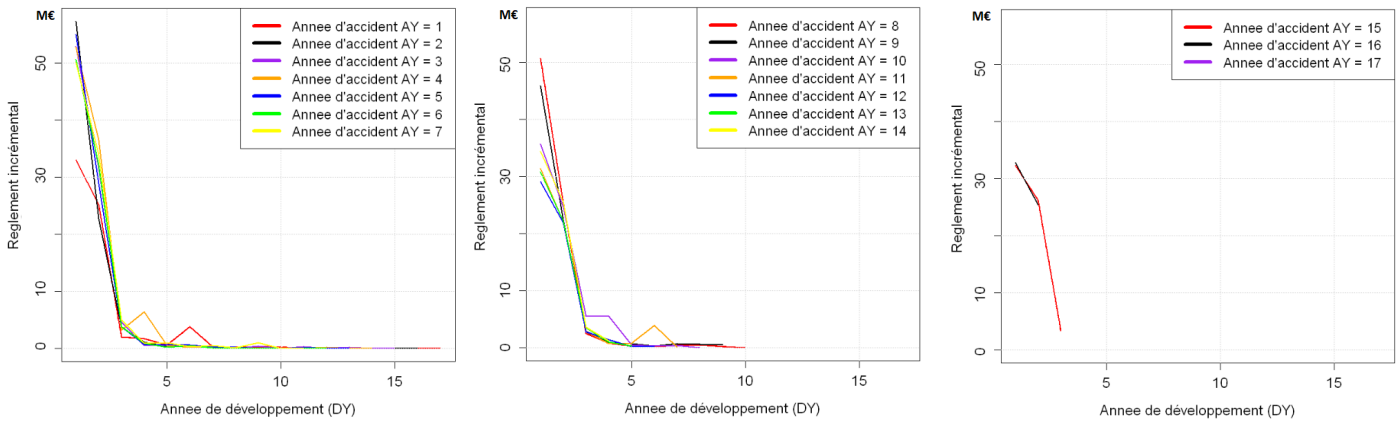

FIGURE 3.9 : Représentation des années d'accident pour la garantie automobile (RC matériel).

Cette garantie ne dispose pas de pic principal au début d'année de développement. En revanche, les flux des règlements décroient rapidement, avec quelques petites volatilités entre la 3ème et la 7ème année de développement. Cet environnement est similaire à l'environnement 1 représenté dans la partie données simulées. La plupart des garanties automobile (responsabilité civile) matériels prennent la même forme que celle représentée dans la figure ***4.10***.

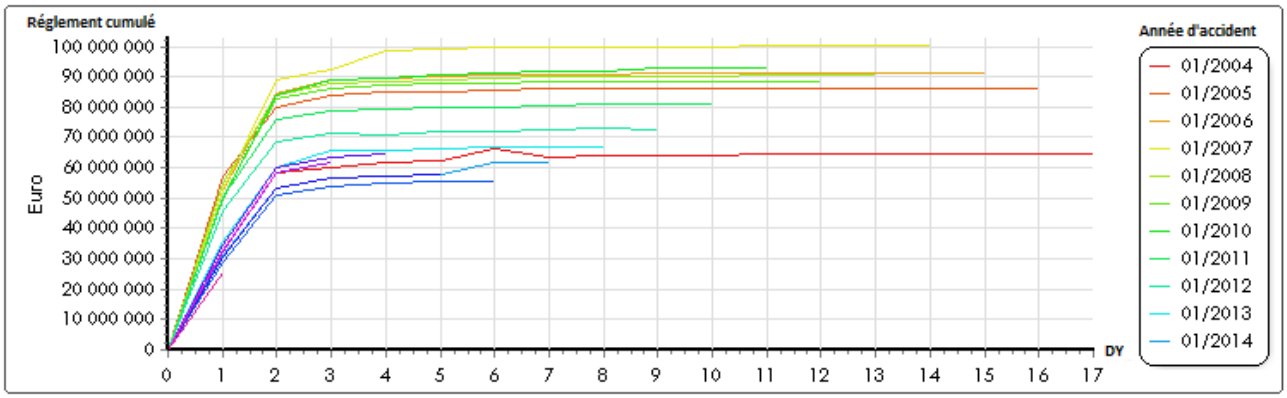

FIGURE 3.10 : Représentation en vision cumulée pour la garantie automobile (RC matériel).

Le graphe ci-dessus représente le réglement cumulé des années d'accident en fonction des années de développement. On remarque que cette branche est à développement court. En effet, les flux des réglements cumules commencent à se stabiliser rapidement dès la 3ème ou la 4ème année de développement.

3.2.2 Automobile (dommages)

Pour les garanties dommages, les biens garantis sont indemnisés selon le principe défini par l'article ***L121-1*** du ***code des assurances***. Les dommages sont évalués de gré à gré ou par expert. L'indemnité est égale au montant des réparations dans la limite de la valeur de remplacement du véhicule assuré ou de sa valeur argus, si celle-ci est plus élevée (déduction faite du prix de l'épave si le véhicule n'est pas réparé). Ces garanties sont soumises à franchise et/ou à plafond. L'existence au contrat de ces garanties induit l'obligation d'assurer le risque catastrophe naturelle (***L125-1***).

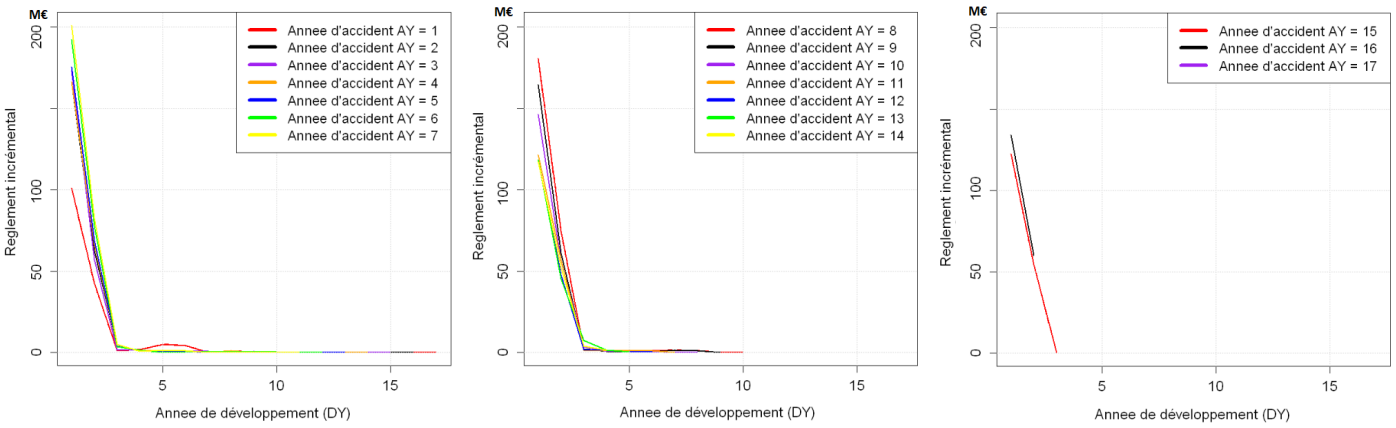

FIGURE 3.11 : Représentation des années d'accident pour la garantie automobile (dommages).

Cet environnement ce ressemble beaucoup à l'environnement précedent et à l'environnement 1, mais sans volatilité remarquable. Les flux des règlements incrémentaux commence à décroitre brutalement dès la première année de développement pour attendre une stabilité en 2ème année de développement. En effet, une indemnisation des dommages est rapide à régler car le montant de cette indemnisation est déterminé rapidement par un examen ou par un expert indépendant.

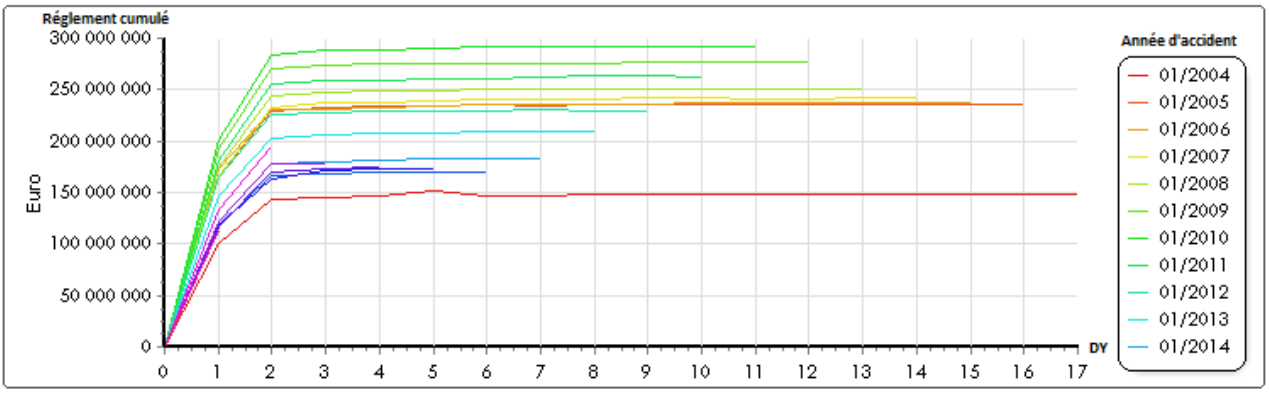

FIGURE 3.12 : Représentation en vision cumulée pour la garantie automobile (dommages).

Le graphe ci-dessus représente le réglement cumulé des années d'accident en fonction des années de développement. On remarque que cette branche est à développement court. En effet, les flux des réglements cumules commencent à se stabiliser rapidement dès la 2ème année de développement.

3.2.3 Responsabilité civile générale

La garantie responsabilité civile générale (RCG) protège si une personne est tenu responsable d'avoir causé involontairement des dommages matériels dans notre cas à un tiers, soit par un produit ou par des activités d'exploitation.

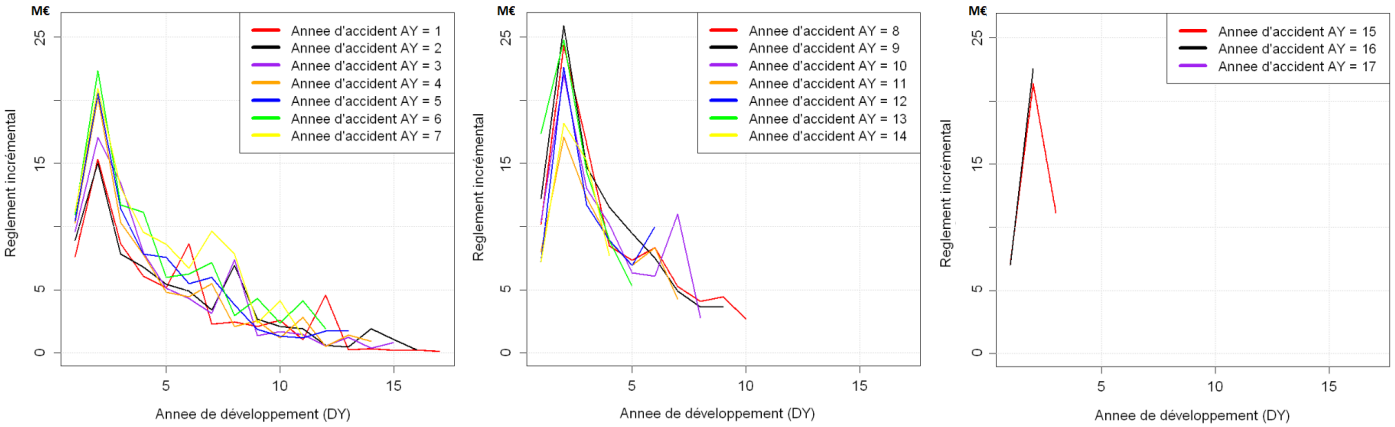

FIGURE 3.13 : Représentation des années d'accident pour la garantie responsabilité civile générale.

Le pic des réglements observé dans la figure ci-dessus à la deuxième année de développement (DY) et la volatilité des incréments ont compliqué ce jeu de données. La compléxité de cette garantie résident dont le fait qu'elle contient des dommages corporels et matériels à la fois, ce qui rend nos données volatiles. Cette garantie est similaire à l'environnement 1 ***mais*** *dans le cas d'une branche à développement long avec une volatilité importante.* Malgré cette complexité, ce type de données est

très fréquent dans la garantie responsabilité civile générale.

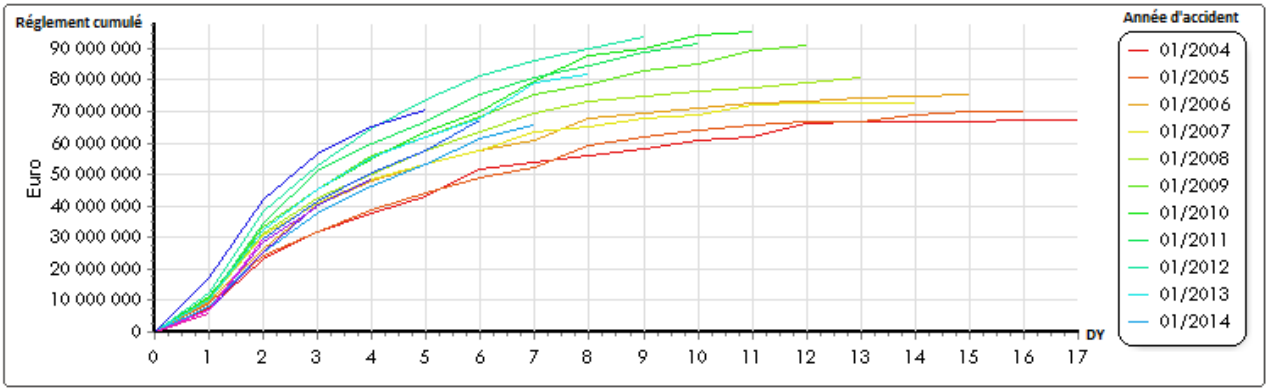

FIGURE 3.14 : Représentation en vision cumulée pour la garantie responsabilité civile générale.

Le graphe ci-dessus représente le réglement cumulé des années d'accident en fonction des années de développement. On remarque que cette branche est longue. En effet, les flux des réglements cumulés commencent à se stabiliser qu'à partir de la 12ème année de développement.

Pour avoir une vue globale, on visualise la surface de chaque garantie afin de bien observer l'évolution des réglements incrémentals, en fonction de l'année d'accident et de développement.

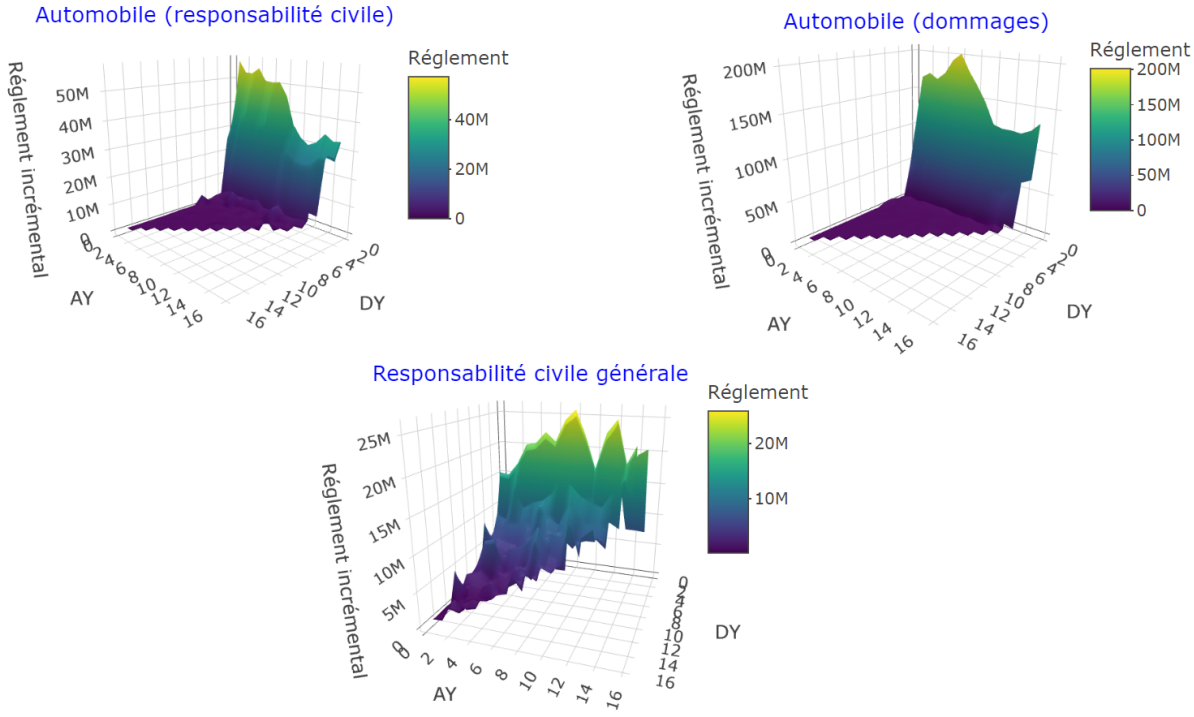

FIGURE 3.15 : Représentation des surfaces des tois garanties.

Le figure ci-dessus permet de nous montrer la difference entre les trois garanties. En effet, pour la garantie automobile (RC matériel) et la la garantie automobile (dommages), il y a une difference au niveau d'ordre de grandeur des règlements, ceux de la première garantie ne dépassent pas 60M€, tandis que ceux de la deuxième garantie atteints les 200M€. Il y a aussi une différence au niveau des

pics entre la 3ème et la 5ème année de développement pour la garantie automobile (résponsabilité civile), tandis que la garantie autimobile (dommages) ne possède pas de pic. En ce qui concerne la garantie résponsabilité civile générale, on remarque une forte volatilté avec une tendance haussière des réglements incrémentaux des années de développement en focntion des années d'accident, ce qui explique le fait que cette garantie est une branche à développement long.

3.3 Avantages et limites de chaque type de données

3.3.1 Données réelles

• **Avantages**

Les données réelles permettent de modéliser les interactions complexes qui peuvent ne pas être saisies par les données simulées. L'ajustement des modèles sur des données réalistes augmentera leur validité.

• **Limites**

Les montants incrémentaux $X_{i,j}$ sont souvent négatifs à cause de la présence de boni de liquidation dans le déroulement de la charge des sinistres, ou l'encaissement de recours en fin de développement, pour les triangles de paiements. Cela ne permet pas d'appliquer les Méthodes de Machine Learning, de Deep Learning et le modèle GLM ODP, car ils requièrent que les montants incrémentaux $X_{i,j}$ soint positifs. En outre, on ne possède pas le triangle inférieur, ce qui ne permet pas de bien tester les projections et comparer les résultats obtenus pour les méthodes utilisées.

3.3.2 Données simulées

• **Avantages**

Conditions de simulation des sinistres très souples, car les distributions sont spécifiées par l'utilisateur, ce qui permet de simuler de nombreux scénarios différents. En plus, les données simulées permettent de comparer les performances des modèles grâce à la disposition des valeurs futures (triangle inférieur), ce qui permet de tester les projections et comparer les résultats obtenus pour les méthodes utilisées.

• **Limites**

Le simulateur SynthETIC ne se calibre pas sur un jeu de données donné, ce qui nécessite une bonne compréhension de la dynamique des sinistres pour spécifier des distributions réalistes, ou un processus de modélisation distinct qui se calibre sur un jeu de données de sinistres externes.

Chapitre 4

Analyse et comparaison de résultats

4.1 Résultats des données simulées

Comme nous l'avons déjà mentionné dans la section ***1.6.3***, les méthodes de Chain Ladder, Mack et GLM ODP donnent les même résultats pour l'estimation centrale, nous allons donc nous restreindre au GLM ODP qui va nous permettre de comparer la distribution des provisions avec celle du MDN.

4.1.1 Environnement 1 : branche de développement court

L'environnement 1 contient des sinistres simples à queue courte dont la composition est homogène pour tous les trimestres d'accident.

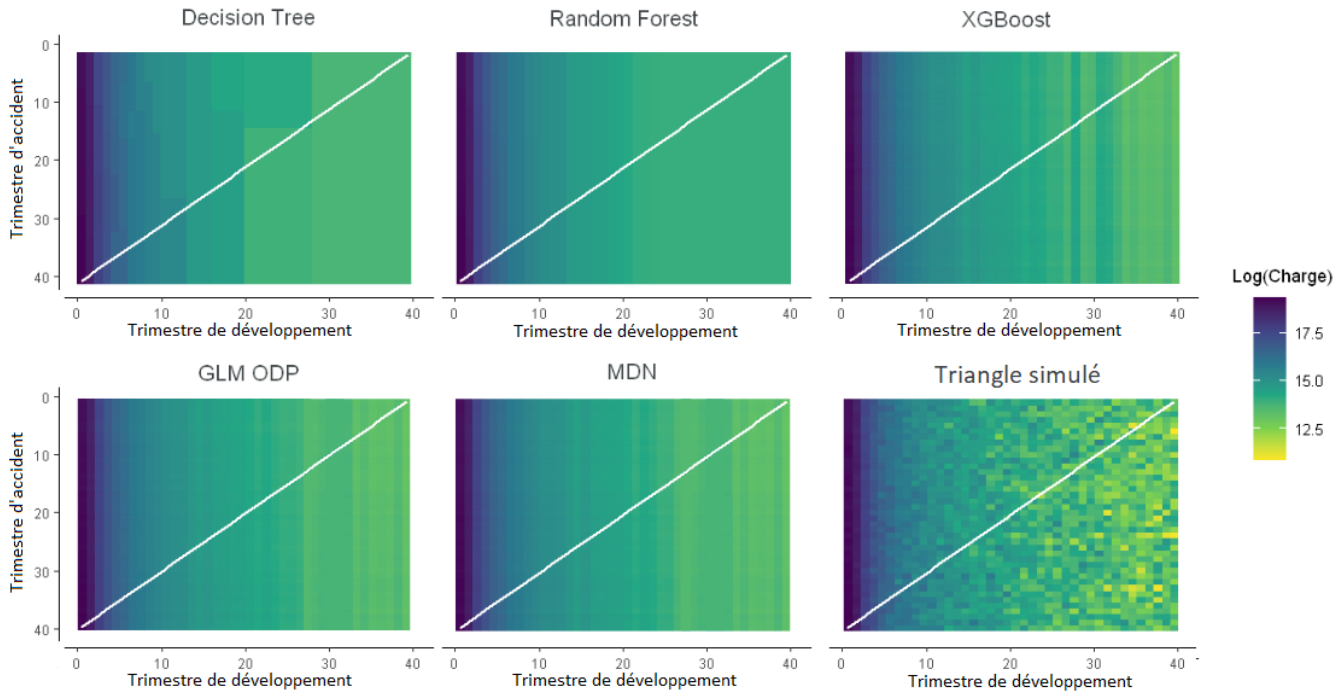

FIGURE 4.1 : Visualisations des charges simulées et prédites pour l'environnement 1.

La figure (***5.1***) représente les données passées et futures (pour les données simulées) et prédites (pour les modèles appliqués), avec ***une ligne diagonale marquant la limite entre le triangle inférieur (passé) et le triangle supérieur (future ou prédiction)***.

Tout d'abord, analysons la charge prédite. Les graphiques ci-dessous représente le log(charge prédite) pour les modèles, ou le log(charge simulée) dans le cas des valeurs simulées. Tous les modèles détectent et projettent l'interaction facilement. On observe la nature en bloc de Decision Tree qui segmente le triangle en parties homogènes. Les prédictions des modèles Random Forest et XGBoost semblent lisses puisqu'ils sont fonction d'un certain nombre d'arbres. Les prédictions du MDN et GLM ODP sont également lisses mais permettent de mieux capturer l'interaction.

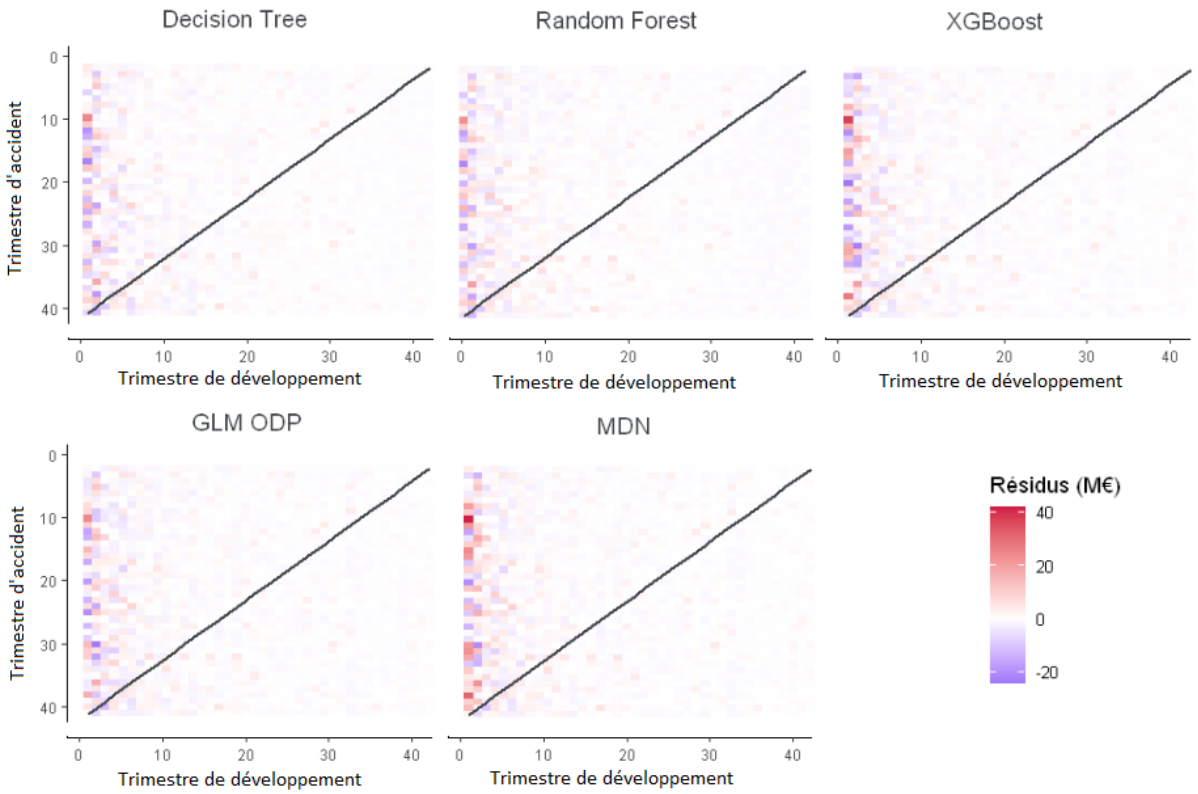

FIGURE 4.2 : Heat Maps des résidus des modèles pour l'environnement 1.

Nous pouvons également examiner la qualité des prédictions des modèles par des cartes thermiques (Heat Maps) des résidus (charge simulée moins charge prédite) representées dans la figure (***5.2***). Cette analyse peut aider à identifier les zones de mauvaise prédiction où il peut y avoir des tendances ou des interactions entre les trimestres d'accident et les trimetres de développement.

Tous les modèles prédisent parfaitement les charges incrémentales après le 2ème trimestre de développement. Pour le 1er et le 2ème trimestre de développement, on remarque quelques résidus importants (au niveau du triangle supérieur i.e la phase d'apprentissage) pour tous les modèles. En revanche, tous les modèles prédisent bien le triangle inférieur, c'est-à-dire pas de résidus remarquables pour la phase de prédiction.

Comparons le score RMSE de prédiction du triangle inférieur de chaque modèle représenté dans le tableau (***5.1***). Le modèle GLM ODP donne le meilleur score RMSE avec un écart relatif très faible par rapport aux modèles MDN et XGBoost et un écart relatif de 6% et de 8% par rapport aux modèles Random Forest et Decision Tree respectivement.

Modèle	Score RMSE (M€)
GLM ODP	1.12
MDN	1.13
XGBoost	1.15
Random Forest	1.19
Decision Tree	1.21

TABLE 4.1 : Score RMSE des modèles pour l'environnement 1.

Par la suite, nous examinons les estimations des provisions des différents modèles. D'après la figure (***5.3***), on observe que tous les modèles suivent parfaitement la tendance des provisions simulées par rapport au trimestre d'accident. Il n'y a pas de volatilité remarquable autour de la valeur simulée. On peut affirmer que les méthodes usuelles (Chain Ladder, Mack, GLM ODP), les méthodes de Machine Learning (Decision tree, Random Forest, XGBoost) et les méthodes de Deep Learning (MDN) ont tous réussi à bien prédire l'évolution des provisions totales en fonction de trimestre d'accident pour l'environnement 1. La seule différence réside dans le fait que le modèle Random Forest sur-estime les provisions comprises entre le 10ème et le 20ème trimestre d'accident.

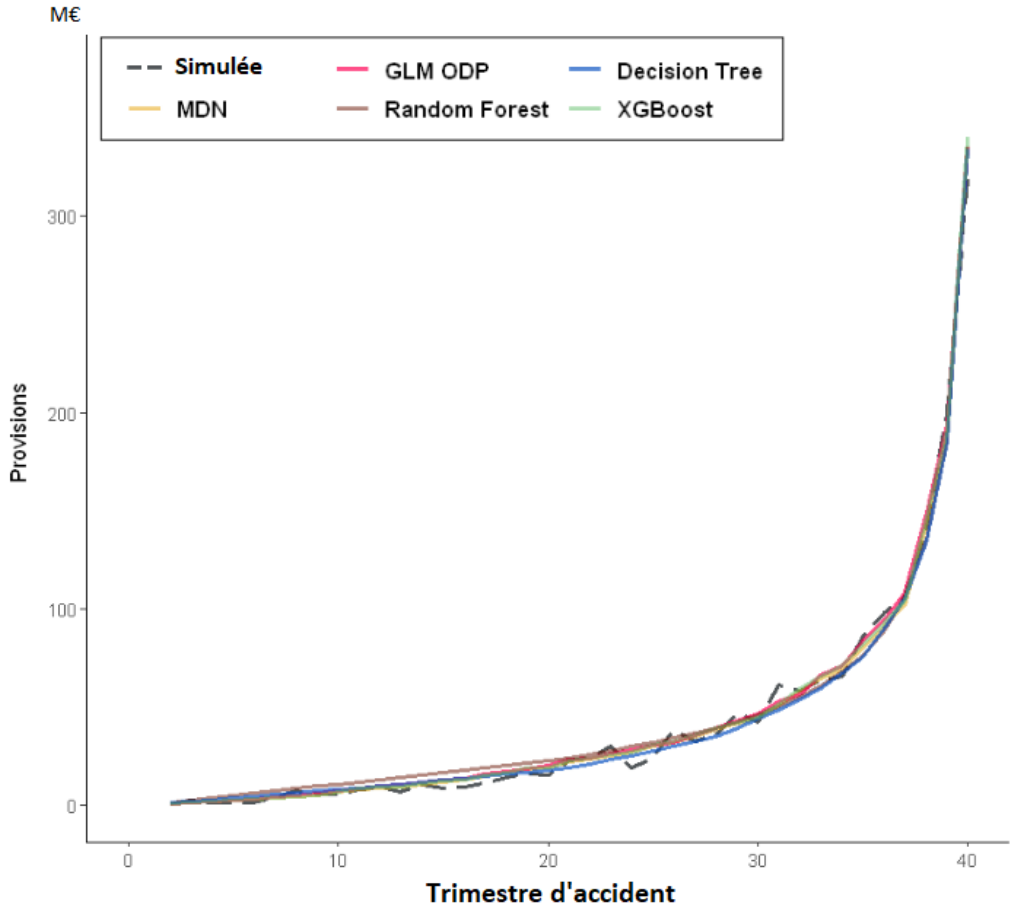

FIGURE 4.3 : Estimations des provisions en fonction du trimestre d'accident pour l'environnement 1.

Le tableau (***5.2***) compare les provisions totales de chaque modèle par rapport à la valeur simulée en utilisant un Back-Testing (ratio de couverture). Le modèle MDN couvre 100.12% des provisions totales, ce qui est plus proche aux provisions simulées. Les modèles XGBoost, Random Forest et GLM

ODP ont des résultats très proches, tandis que Decision Tree sous-estime les provisions totales de 1.87%. Enfin, tous les modèles ont réussi à bien prédire les provisions totales.

Modèle	Provisions (M€)	Ratio (%)
Simulée	1652.47	100.00
MDN	1654.43	100.12
XGBoost	1688.19	102.16
GLM ODP	1707.85	103.35
Random Forest	1722.95	104.27
Decision Tree	1621.52	98.13

TABLE 4.2 : Provisions totales et Backtesting pour l'environnement 1.

Ensuite, nous comparerons les densités des provisions totales, le SCR_{Ultime} et le CoV (*Coefficient of Variation*) du GLM ODP et du MDN par rapport à la densité simulée.

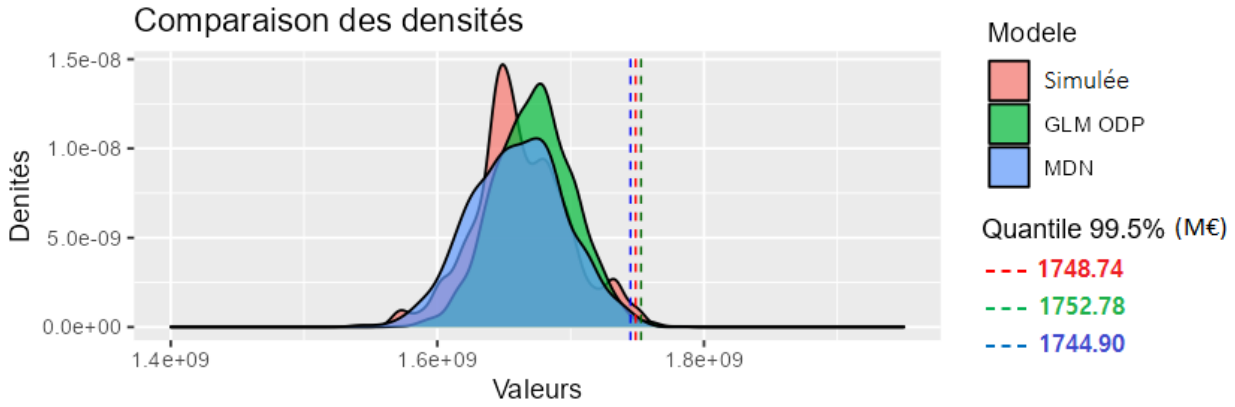

FIGURE 4.4 : Comparaison des densités de provisions pour l'environnement 1.

Les densités du MDN et GLM ODP sont très proches de la densité simulée. La précision de la distribution du MDN est due à sa capacité à modéliser des tendances complexes dans les données ainsi qu'à sa capacité à ajuster une distribution plus flexible (modèle de mélange gaussien). La distribution du GLM ODP obtenue par la méthode Boostrap présentée dans la section ***1.6.3*** donne aussi un résultat très satisfaisant dans le cas d'un environnement simple. La densité simulée est obtenue en générant *200 triangles* du même environnement et qui suivent les mêmes hypothèses.

Modèle	SCR_{Ultime} (M€)	CoV (%)
Simulée	96.27	21.11
MDN	90.71	21.57
GLM ODP	44.93	17.74

TABLE 4.3 : SCR_{Ultime} et CoV pour l'environnement 1.

Le modèle MDN est le plus précis des trois en matière de SCR_{Ultime} et CoV. En revanche, le CoV du GLM ODP est aussi proche mais le SCR_{Ultime} est considérablement sous-estimé.

Les méthodes de Machine Learning et de Deep Learning sont plus précises que les méthodes usuelles. Néanmoins, d'après le tableau (***5.4***), leurs traitements sont chronophages et énergivores

(pour le hyper-paramètres et l'apprentissage) par rapport aux méthodes usuelles, qui donnent les résultats instantanément.

Modèle	*Temps écoulé pour le hyper-paramètrage*	*Temps écoulé pour l'apprentissage*
GLM ODP / Chain Ladder / Mack	-	instantané
Decision Tree	2 min 45 sec	30 msec
Random Forest	14 min 31 sec	123 msec
XGBoost	1 h 41 min	890 msec
MDN	6 h 13 min	9 min 78 msec

TABLE 4.4 : Comparaison temporelle des modèles pour l'environnement 1.

Pour un environnement simple comme celui-ci, l'analyse coût-bénéfice permet de tirer la conclusion suivante : il est préférable d'utiliser les méthodes usuelles à celles de Machine Learning ou Deep Learning pour augmenter le gain de temps au détriment d'une légère baisse de précision parfois négligeable.

4.1.2 Environnement 2 : variation de la cadence des sinistres

Dans l'environnement 2, les caractéristiques des sinistres changent progressivement de nature en agrégeant deux branches (branche à développement long et une branche à développement court). En effet, la proportion des sinistres longs diminue, tandis que la proportion des sinistres courts augmente.

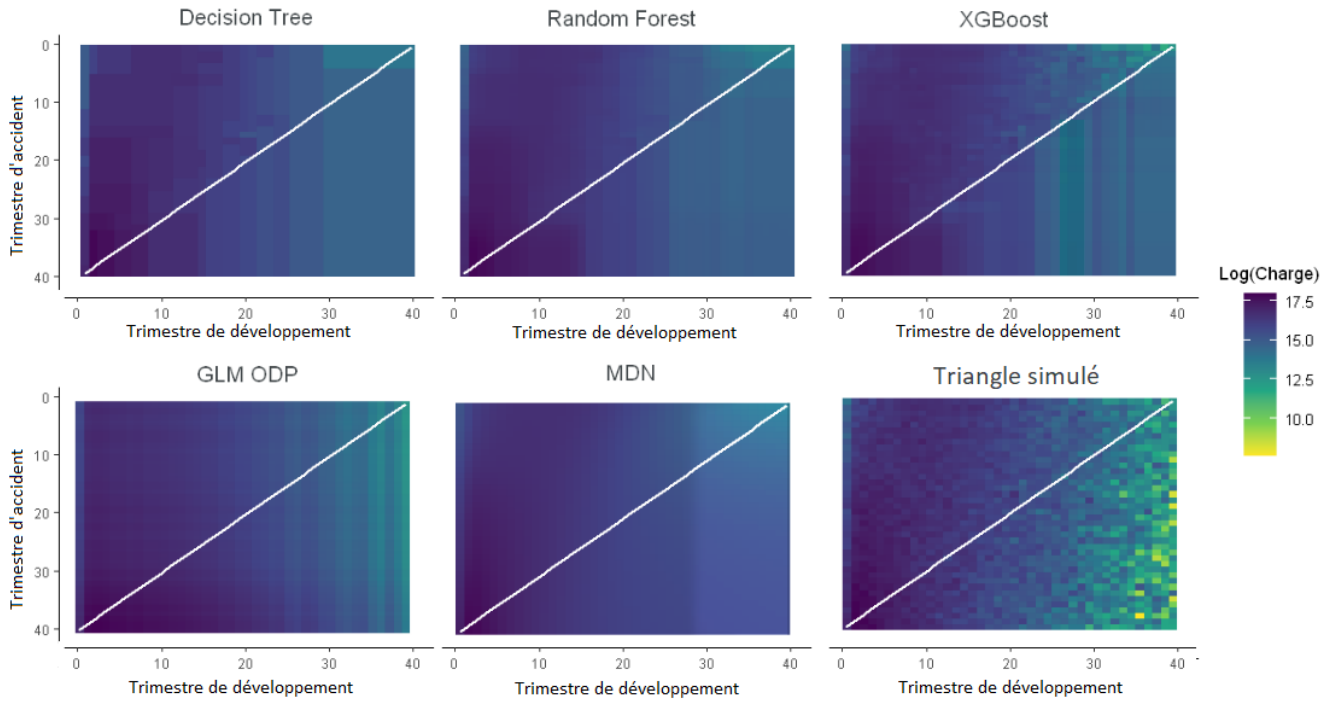

FIGURE 4.5 : Visualisations des charges simulées et prédites pour l'environnement 2.

Les graphiques ci-dessus montrent que les modèles ont des difficultés à détecter et projeter l'interaction des derniers trimestres de développement. On observe la nature en bloc de Decision Tree qui

segmente le triangle en parties homogènes. Les prédictions des modèles Random Forest et XGBoost semblent lisses puisqu'ils sont fonction d'un certain nombre d'arbres. La prédiction du MDN est plus lisse, mais ne capture pas la variation de la cadence des sinistres parfaitement. On observe aussi l'effet des coefficients de développement du GLM ODP.

Le Heat Maps représenté dans la figure (***5.6***) montre que les modèles MDN et XGBoost ne possèdent pas de région qui représente des défauts de prédiction. Les modèles Decision Tree, Random Forest et GLM ODP possèdent une région où les résidus sont importants. Ce défaut est expliqué par la variation de la cadence des sinistres.

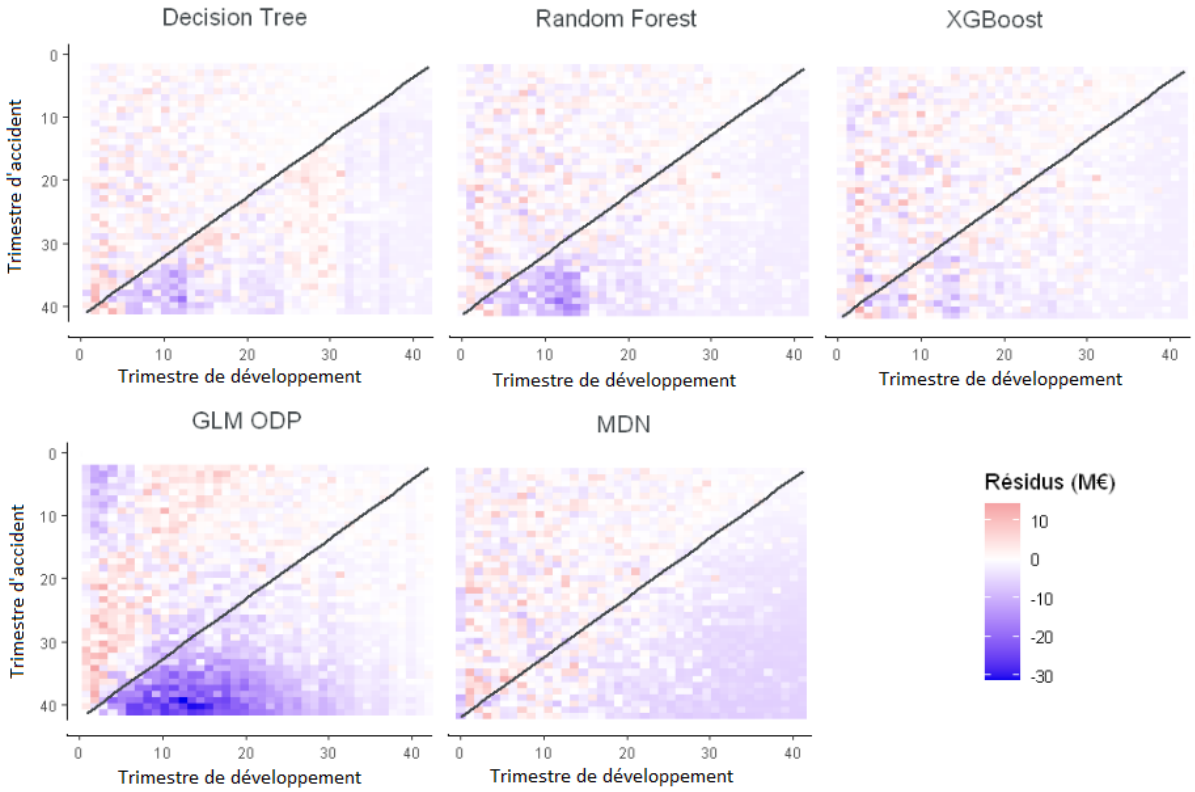

FIGURE 4.6 : Heat Maps des modèles pour l'environnement 2.

Comparons le score RMSE de prédiction du triangle inférieur de chaque modèle :

Modèle	Score RMSE (M€)
MDN	2.80
XGBoost	2.93
Decision Tree	3.21
Random Forest	3.66
GLM ODP	8.63

TABLE 4.5 : Score RMSE des modèles pour l'environnement 2.

Le modèle MDN donne le meilleur score RMSE, avec un écart relatif de 4.64% par rapport au modèle XGBoost et un écart relatif de 14.64% et de 30.71% par rapport aux modèles Decision Tree et Random Forest respectivement. Il s'éloigne considérablement du score de GLM ODP (un écart relatif de 208.21%).

D'après la figure (***5.7***), on observe qu'avant le 22ème trimestre d'accident tous les modèles suivent parfaitement la tendance des provisions simulées par rapport au trimestre d'accident. Après ce trimestre d'accident, le modèle GLM ODP commence à diverger et s'éloigner des provisions simulées. Les modèles MDN et Decision Tree donnent des résultats plus précises. Une fois ce trimestre d'accident dépassé, les modèles Random Forest et XGBoost commencent à s'éloigner des provisions totales simulées suites à la variation progressive des sinistres, mais sans diverger. Enfin, les méthodes de Machine Learning et Deep Learning ont pu prédire l'évolution des provisions totales sans divergence remarquable. En revanche le modèle GLM ODP a échoué sur l'évolution des provisions totales.

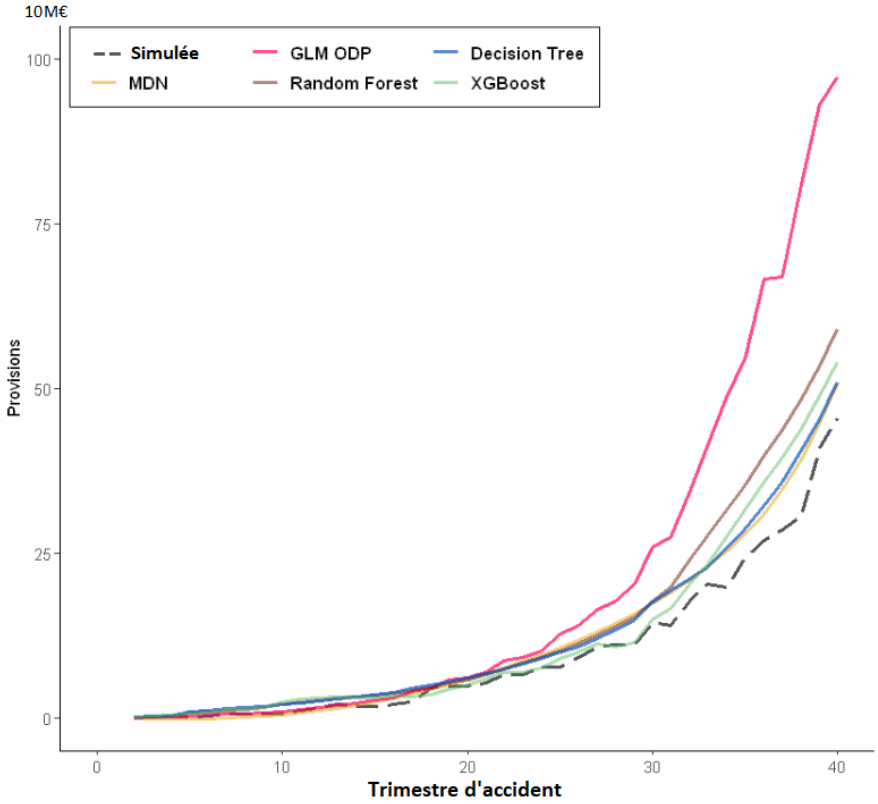

FIGURE 4.7 : Estimations des provisions en fonction du trimestre d'accident pour l'environnement 2.

Le tableau (***5.10***) compare les provisions totales de chaque modèle par rapport à la valeur simulée en utilisant un Back-Testing.

Modèle	Provisions (M€)	Ratio (%)
Simulée	3889.61	100
MDN	4613.36	118.60
XGBoost	4796.22	123.31
Decision Tree	4815.85	123.81
Random Forest	5418.03	139.29
GLM ODP	7905.54	203.25

TABLE 4.6 : Provisions totales et Backtesting pour l'environnement 2.

Le modèle MDN couvre 118.60% des provisions simulées, c'est appréciable pour un environnement complexe. Les modèles XGBoost et Decision Tree ont des résultats très proches et couvrent respectivement 123.31% et 123.81%, tandis que Random Forest couvre moins de provisions. Le modèle GLM

ODP couvre le double de ces provisions totales ce qui est problèmatique en provisionnement.

Ensuite, nous comparerons les densités des provisions totales, le SCR_{Ultime} et le CoV du GLM ODP et du MDN par rapport à la densité simulée.

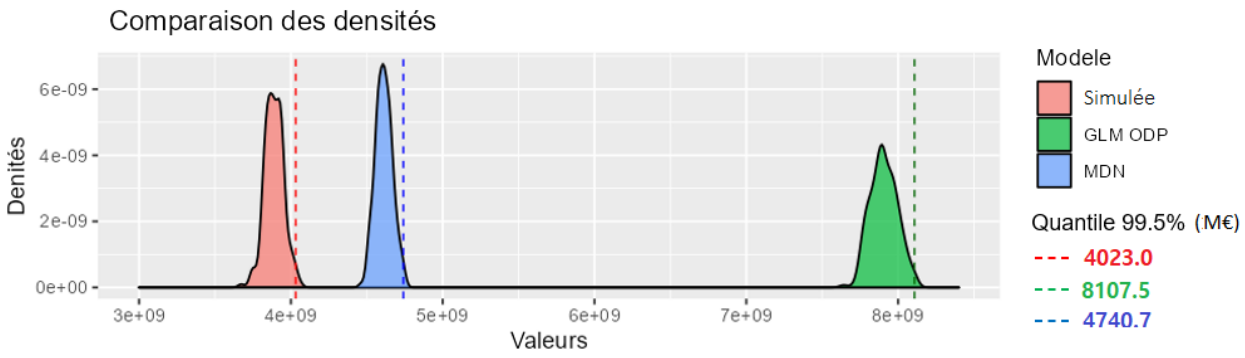

FIGURE 4.8 : Comparaison des densités des provisions totales pour l'environnement 2.

La densité du MDN ne s'éloigne pas de la densité des données simulées. Par contre, la densité du GLM ODP s'éloigne beaucoup de la densité des données simulées. La précision de la distribution du MDN est due à sa capacité à modéliser des tendances complexes dans les données ainsi qu'à sa capacité à ajuster une distribution plus flexible. La distribution du GLM ODP obtenue par la méthode Boostrap présentée dans la section ***1.6.3*** ne réussi pas à se rapprocher de la densité des données simulées, puisque l'environnement 2 est complexe. La densité simulée est obtenue en générant *200 triangles* de même environnement qui suivent les mêmes hypothèses.

Modèle	SCR_{Ultime} (M€)	CoV (%)
Simulée	142.47	15.47
MDN	127.41	12.69
GLM ODP	201.92	11.55

TABLE 4.7 : SCR_{Ultime} et CoV pour l'environnement 2.

Le MDN est plus précis en termes de SCR_{Ultime} et CoV. En revanche, le CoV du GLM ODP est aussi proche mais son SCR_{Ultime} est très important que celui des données simulées.

Modèle	***Temps écoulé pour le hyper-paramètrage***	***Temps écoulé pour l'apprentissage***
GLM ODP / Chain Ladder / Mack	-	instantané
Decision Tree	3 min 25 sec	33 msec
Random Forest	1 h 10 min	580 msec
XGBoost	2 h 1 min	1 sec 516 msec
MDN	7 h 32 min	12 min 26 msec

TABLE 4.8 : Comparaison temporelle des modèles pour l'environnement 2.

Pour un environnement complexe comme celui-là, l'analyse coût-bénéfice permet de tirer la conclusion suivante : Si on veut gagner le temps, on va avoir un résultat moins bon avec GLM ODP. Par contre, si on utilise le résultat de Decision Tree, on va optimiser le couple coût-bénéfice, mais on ne va pas avoir la densité des provisions pour calculer la CoV et le SCR. Dans ce cas, Il est préférable d'avoir des résultats plus précis au détriment du temps plutôt qu'une précision inférieure avec un traitement plus rapide. Pour cet environnement, l'utilisation du MDN est privilégiée.

4.1.3 Environnement 3 : branche de développement long avec choc d'inflation

Dans l'environnement 3, l'inflation trimestrielle superposée passe instantanément de 0% à 10%, après le 30ème trimestre. L'inflation trimestrielle de base est fixée à 2%.

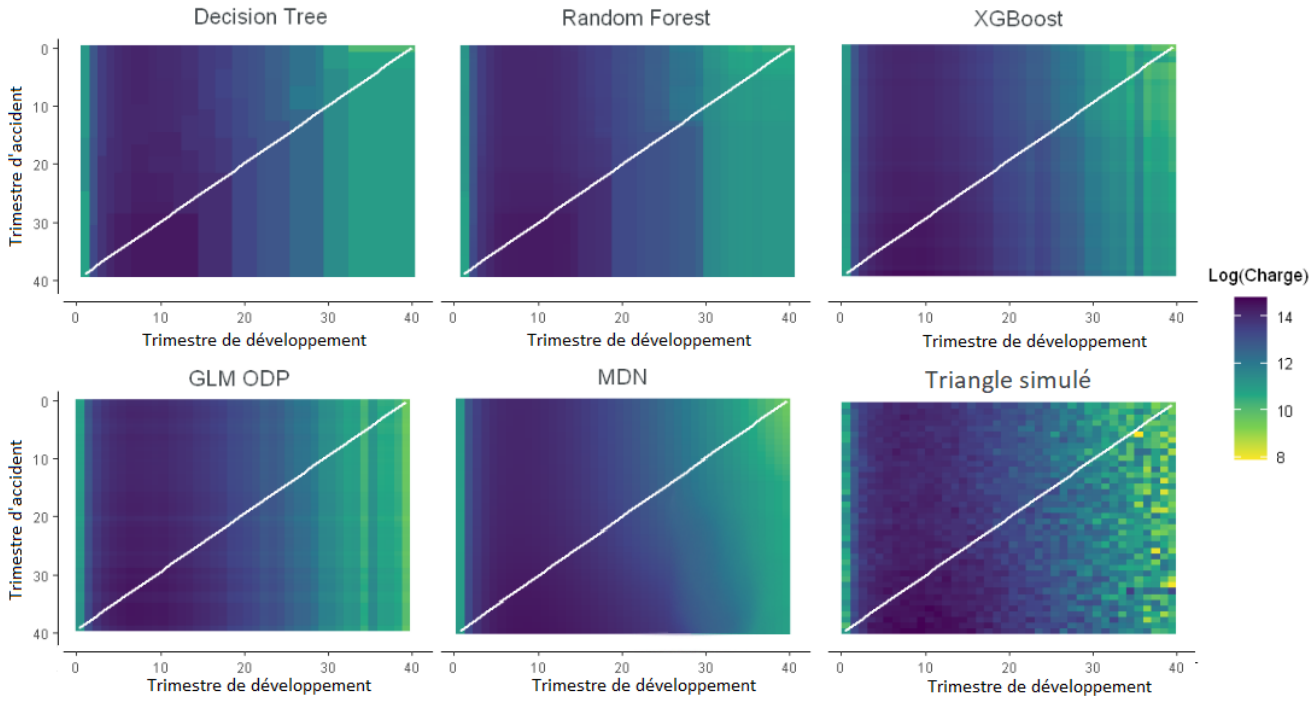

FIGURE 4.9 : Visualisations des charges simulées et prédites pour l'environnement 3.

Les graphiques ci-dessus montrent que les modèles ont quelques difficultés pour détecter et projeter l'interaction de l'inflation des derniers trimestres de développement. On observe la nature en bloc de Decision Tree qui segmente le triangle en parties homogènes. Les prédictions des modèles Random Forest et XGBoost semblent lisses puisqu'ils sont fonction d'un certain nombre d'arbres. La prédiction du MDN est la plus lisse et capture partiellement le choc de l'inflation. On observe aussi l'effet des coefficients de développement du GLM ODP.

D'après la figure (***5.10***), tous les modèles prédisent parfaitement les charges incrémentales avant le 30ème trimestre d'accident. Après ce trimestre d'accident, on commence à remarquer quelques différences de prédiction pour tous les modèles. En effet, Decision Tree et Random Forest predisent les charges en utilisant la prédiction par bloc, ce qui a sous-estimé les charges du triangle inférieur, sans detercter l'effet de l'inflation. Le GLM ODP à son tour n'a pas pu capturer l'effet de cette inflation. La prédiction de XGBoost est beaucoup mieux que les modèles précedents. Le MDN a réussi à prédire l'inflation d'une manière partielle en sur-estimant les charges après le 20ème trimestre de développement. Enfin, tous les modèles ont des défauts de prédiction causés par le choc d'inflation,

mais le modèle MDN semble être le plus performant.

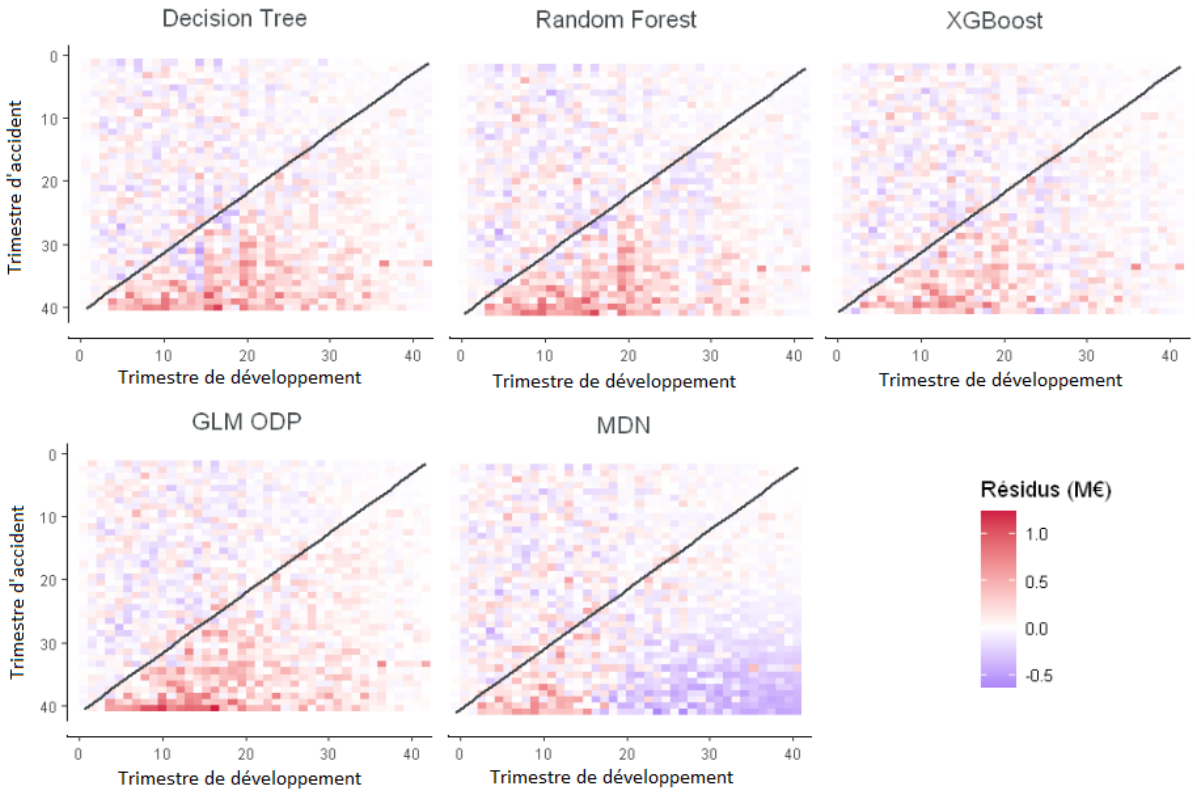

FIGURE 4.10 : Heat Maps des modèles pour l'environnement 3.

Ci-dessous, une comparaison est exposée sur le score RMSE de prédiction du triangle inférieur de chaque modèle.

Modèle	Score RMSE (M€)
MDN	0.185
XGBoost	0.214
Decision Tree	0.227
Random Forest	0.231
GLM ODP	0.241

TABLE 4.9 : Score RMSE des modèles pour l'environnement 3.

Le modèle MDN donne le meilleur score RMSE, avec un écart relatif de 15.67% par rapport au modèle XGBoost et un écart relatif de 22.70% et de 24.86% par rapport aux modèles Decision Tree et Random Forest respectivement. Il s'éloigne beaucoup du score de GLM ODP avec un écart relatif de 30.27%.

Par la suite, nous examinerons les estimations de provisions des différents modèles. D'après la figure (***5.11***), on observe que tous les modèles suivent subjectivement la tendance des provisions simulées par rapport au trimestre d'accident avant le 30ème trimestre d'accident. Après ce trimestre d'accident, l'effet de l'inflation influe sur le comportement des modèles. En effet, le modèle GLM ODP commence à sous-estimer les provisions et à s'éloigner des provisions simulées, jusqu'à une chute totale à partir du 37ème trimestre d'accident. Les modèles Decision Tree et Random Forest ont pu suivre la tendance des provisions simulées, mais en s'éloignant progressivement de cette dernière. Le XGBoost

a beaucoup mieux suivi la tendance des provisions simulées en sous-estimant les provisions. Le Modèle MDN est le plus performant. Il a réussi à suivre l'évolution des provisions totales, en sur-estimant celles-ci.

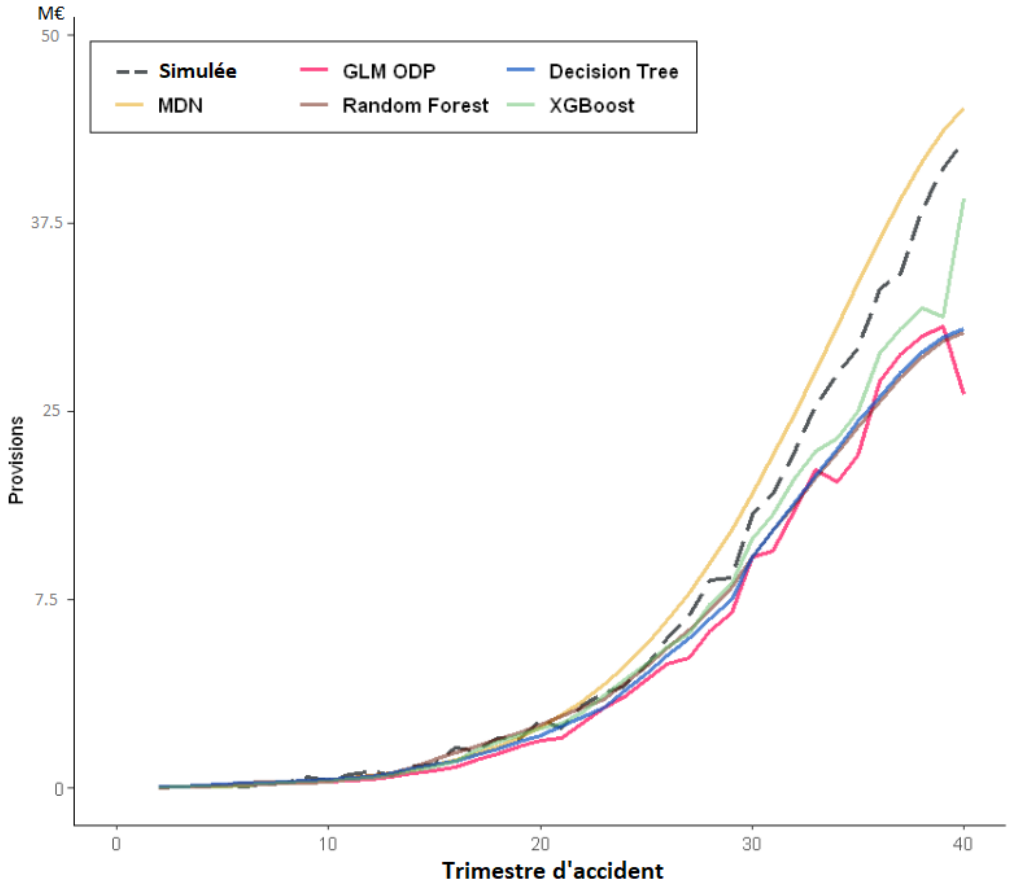

FIGURE 4.11 : Estimations des provisions en fonction du trimestre d'accident pour l'environnement 3.

Le tableau (***5.10***) compare les provisions totales de chaque modèle par rapport à la valeur simulée en utilisant un Back-Testing.

Modèle	Provisions (M€)	Ratio (%)
Simulée	435.65	100
MDN	476.14	109.29
XGBoost	384.12	88.17
Random Forest	359.36	82.49
Decision Tree	353.05	81.04
GLM ODP	337.32	77.43

TABLE 4.10 : Provisions totales et Backtesting pour l'environnement 3.

Le modèle MDN couvre 109.29% des provisions simulées, ce qui est satisfaisant pour un environnement choqué par l'inflation. Les modèles Random Forest et Decision Tree couvrent 81.04% et 82.49% respectivement des provisions simulées, tandis que le GLM ODP couvre moins de provisions. Le XGBoost couvre 88.17% des provisions simulées. Les méthodes usuelles et les méthodes de Machine Learning sous-estiment les provisions totales, tandis que le MDN couvre plus de provisions en restant proche des provisions simulées.

Ensuite, nous comparerons les densités des provisions totales, le SCR_{Ultime} et le CoV du GLM ODP et du MDN par rapport à la densité simulée.

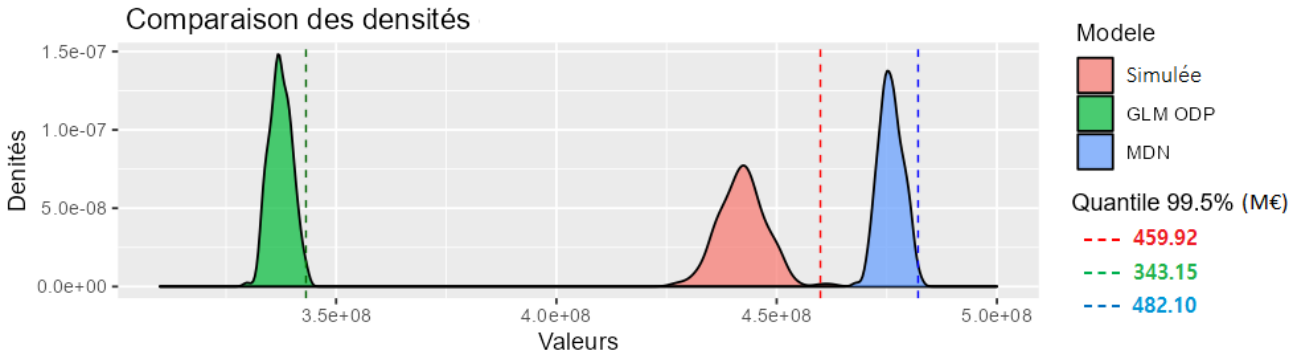

FIGURE 4.12 : Comparaison des densités des provisions totales pour l'environnement 3.

La densité du MDN ne s'éloigne pas de la densité des données simulées. Par contre, la densité du MDN ODP s'éloigne considérablement de la densité des données simulées. La précision de la distribution du MDN est due à sa capacité à modéliser des tendances complexes dans les données, ainsi qu'à sa capacité à ajuster une distribution plus flexible. La distribution du GLM ODP obtenue par la méthode Boostrap présentée dans la section ***1.6.3***, ne réussit pas à se rapprocher de la densité des données simulées. En revanche, il est claire que la forme de la densité des données simulées est differente de celle du MDN et GLM ODP, ce qui traduit la complexité de l'environnement choqué par l'inflation. La densité simulée est obtenue en générant *200 triangles* de même environnement qui suivent les mêmes hypothèses.

Modèle	SCR_{Ultime} (M€)	CoV (%)
Simulée	24.27	13.87
MDN	5.96	9.94
GLM ODP	5.83	8.71

TABLE 4.11 : SCR_{Ultime} et CoV pour l'environnement 3.

Dans cet environnement, le MDN et le GLM ODP ne parviennentt pas à se rapprocher du SCR_{Ultime} et de la CoV des données simulées, le choc de l'inflation influe beaucoup sur le SCR_{Ultime}. En effet, il y a un ecart très remarquable entre les SCR_{Ultime} des modèles MDN et GLM ODP et le SCR_{Ultime} des données simulées. Cependant, la survenance d'un tel choc d'inflation est rare dans la réalité.

Les méthodes de Machine Learning et de Deep Learning sont plus précises que les méthodes usuelles, mais elles requièrent un temps important en vue de calibrer les hyper-paramètres et pour l'apprentissage par rapport aux méthodes usuelles, qui donnent les résultats instantanément.

Pour un environnement choqué brutalement par l'inflation, l'analyse coût-bénéfice permet de tirer la conclusion suivante : Si on veut gagner du temps alors on aura un résultat moins bon avec GLM ODP. Par contre, si on utilise le résultat de XGBoost, on va optimiser le couple coût-bénéfice, mais on ne va pas avoir la densité pour calculer la CoV et le SCR. Dans ce cas, il est préférable d'avoir des résultats plus précis au détriment du temps plutôt qu'une précision inférieure avec un traitement plus rapide. Pour cet environnement, l'utilisation du MDN est privilégié.

Modèle	*Temps écoulé pour le hyper-paramètrage*	*Temps écoulé pour l'apprentissage*
GLM ODP / Chain Ladder / Mack	-	instantané
Decision Tree	26 sec 276 msec	54 msec
Random Forest	37 min 42 sec	834 msec
XGBoost	4 h 8 min	284 msec
MDN	7 h 42 min	16 min 13 msec

TABLE 4.12 : Comparaison temporelle des modèles pour l'environnement 3.

4.1.4 Environnement 4 : branche de développement long à forte volatilité

Dans l'environnement 4, la volatilité des sinistres est très élevée avec une sévérité très volatile. C'est un environnement rare en assurance et difficile à traiter, car les hypothèses de Chain-Ladder ne sont pas verifiées.

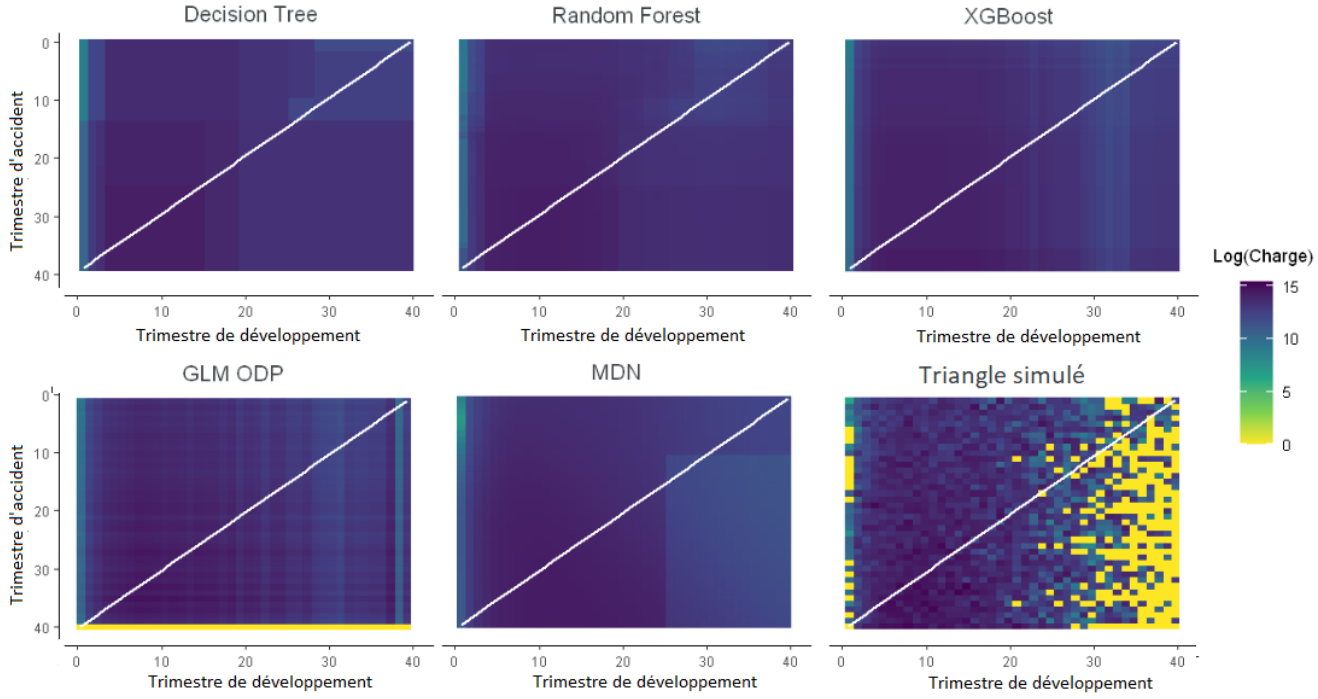

FIGURE 4.13 : Visualisations des charges simulées et prédites pour l'environnement 4.

Les graphiques ci-dessus montrent que les modèles ont des difficultés à détecter et projeter l'interaction des derniers trimestres de développement. On observe la nature en bloc de Decision Tree qui segmente le triangle en parties homogènes. Les prédictions des modèles Random Forest et XGBoost semblent lisses puisqu'ils sont fonction d'un certain nombre d'arbres. La prédiction du MDN capture partiellement la volatilité des sinistres, mais il est incapable de prédire les zones où les charges sont

quasi-nulles. On observe aussi l'effet des coefficients de développement du modèle GLM ODP, ce dernier est incapable de prédire les charges du dernier trimestre d'accident (il ne prédit que les charges nulles).

D'après la figure (***5.14***), tous les modèles ne prédisent pas parfaitement les charges incrémentales. Tous les modèles ont des défauts de prédiction causés par la forte volatilité. Le modèle GLM ODP ne réussit pas à prédire le dernier trimestre d'accident et le 38ème trimestre de développement de chaque trimestre d'accident. Les modèles de Machine Learning et de Deep Learning ont tous des résidus importants au niveau du triangle inférieur, mais réussissent à prédire partiellement le dernier trimestre d'accident (contrairement au modèle GLM ODP).

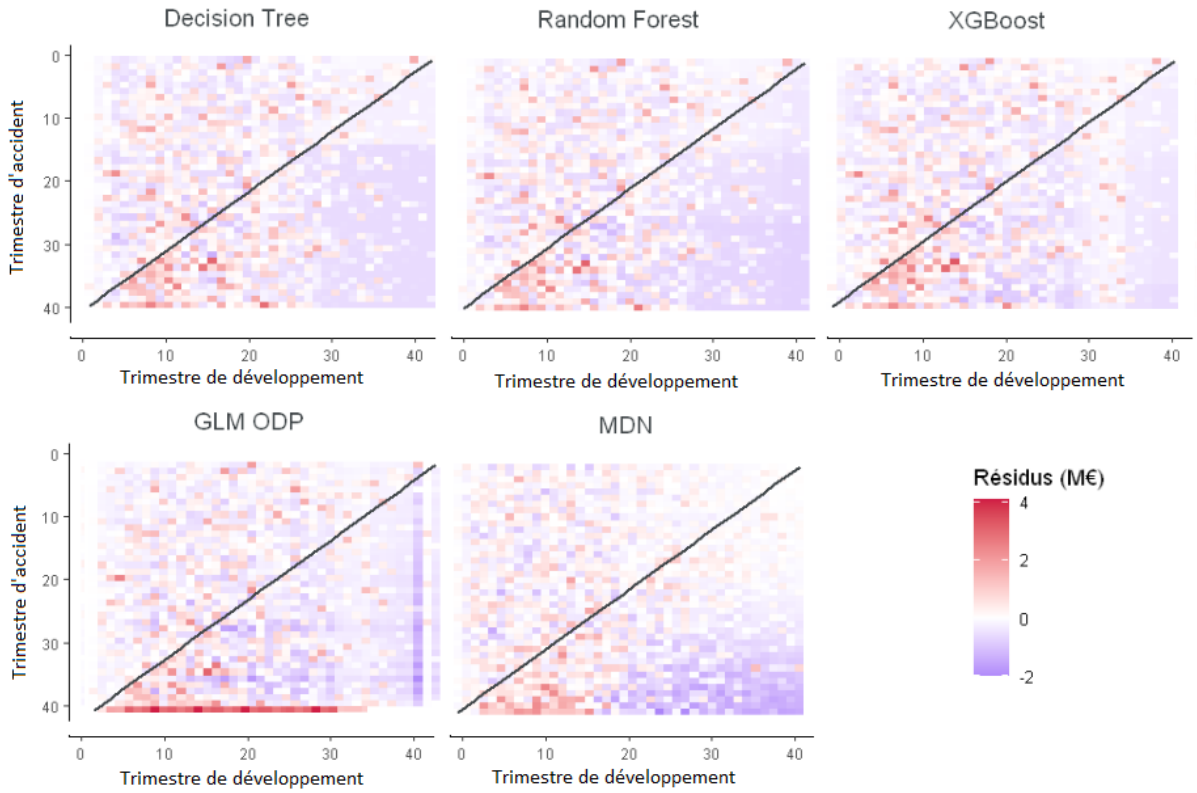

FIGURE 4.14 : Heat Maps des modèles pour l'environnement 4.

Comparons le score RMSE de prédiction du triangle inférieur de chaque modèle :

Modèle	Score RMSE (M€)
MDN	0.582
XGBoost	0.601
Decision Tree	0.640
Random Forest	0.658
GLM ODP	0.741

TABLE 4.13 : Score RMSE des modèles pour l'environnement 4.

Le modèle MDN donne le meilleur score RMSE, avec un écart relatif de 3.26% par rapport au XGBoost et un écart relatif de 9.96% et 13.05% par rapport aux modèles Decision Tree et Random Forest respectivement. Il s'éloigne beaucoup du score de GLM ODP avec un écart relatif de 27.31%.

Par la suite, nous examinerons les estimations des provisions des différents modèles. d'après la figure (***5.19***), on observe qu'aucun modèle n'a pu suivre parfaitement l'évolution des provisions simulées par rapport au trimestre d'accident. Le modèle GLM ODP sous-estime beaucoup les provisions après le 38ème trimestre d'accident. Les modèles de Machine Learning et de Deep Learning prédisent des valeurs moyennes et ne réussissent pas à suivre la progression de la volatilité des provisions simulées.

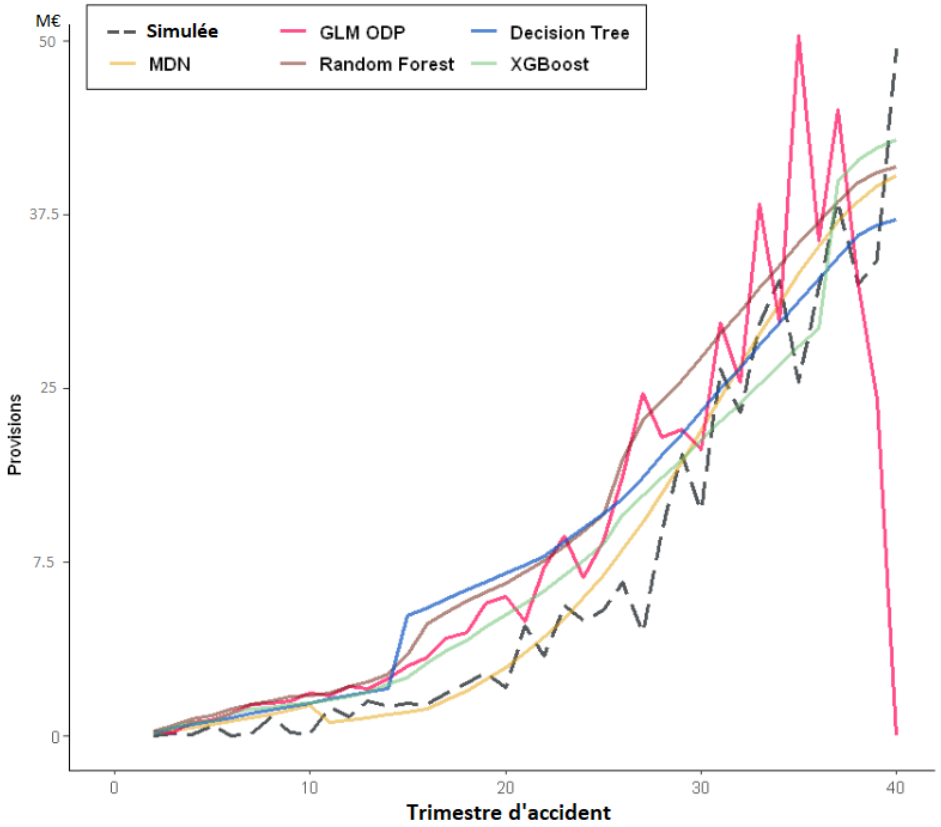

FIGURE 4.15 : Estimations des provisions en fonction du trimestre d'accident pour l'environnement 4.

Le tableau (***5.14***) compare les provisons totales de chaque modèle par rapport à la valeur simulée en utilisant un Back-Testing.

Modèle	Provisions (M€)	Ratio (%)
Simulée	463.78	100.00
MDN	497.70	107.31
XGBoost	536.10	115.59
GLM ODP	550.55	118.71
Random Forest	572.26	123.39
Decision Tree	629.25	135.68

TABLE 4.14 : Provisions totales et Backtesting pour l'environnement 4.

Le modèle MDN couvre 107.31% des provisions simulées, ce qui est satisfaisant pour un environnement très volatile. Les modèles XGBoost et GLM ODP ont des résultats très proches, mais le modèles GLM ODP donne une image biaisée des provisions totales puisqu'il sur-estime les provisions avant le 37ème trimestre d'accident et prédit des provisions nulles au dernier trimestre d'accident. Les modèles Random Forest et Decision Tree sur-estime les provisions totales de 23.39% et 35.68% respectivement.

Ensuite, nous comparerons les densités des provisions totales, le SCR_{Ultime} et le CoV du GLM ODP et du MDN par rapport à la densité simulée.

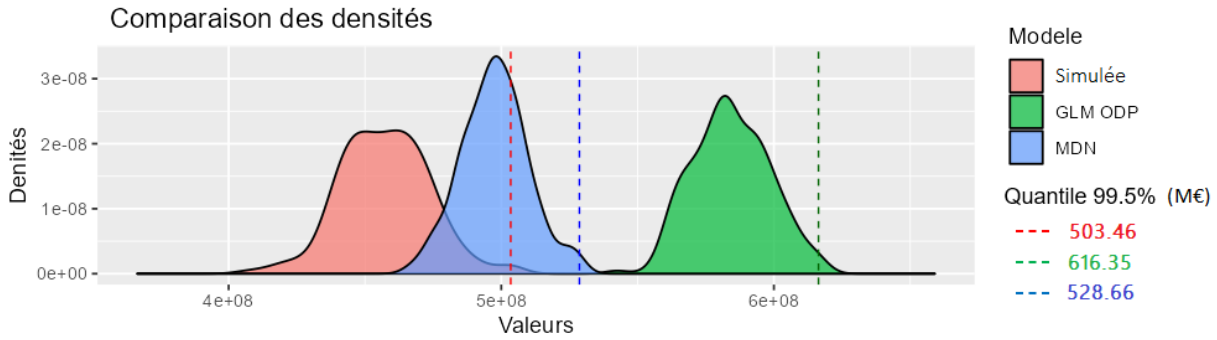

FIGURE 4.16 : Comparaison des densités des provisions totales pour l'environnement 4.

La densité du MDN ne s'éloigne pas de celle des données simulées. Par contre, la densité du GLM ODP s'éloigne un peu de la densité des données simulées. La précision de la distribution du MDN est due à sa capacité à modéliser des tendances complexes dans les données, ainsi qu'à sa capacité à ajuster une distribution plus flexible. La distribution du GLM ODP obtenue par la méthode Boostrap présentée dans la section ***1.6.3***, ne réussit pas à se rapprocher de la densité des données simulées. La densité simulée est obtenue en générant *200 triangles* de même environnement qui suivent les mêmes hypothèses.

Modèle	SCR_{Ultime} (M€)	CoV (%)
Simulée	39.68	38.58
MDN	30.97	24.42
GLM ODP	65.80	24.55

TABLE 4.15 : SCR_{Ultime} et CoV pour l'environnement 4.

Dans cet environnement, le MDN a réussi de se rapprocher partiellement au SCR_{Ultime} des données simulées, mais ne réussi pas de se rapprocher de la CoV des données simulées (une différence importante).Le modèle GLM ODP ne réussi pas à se rapprocher du SCR_{Ultime} (une grande sur-estimation) et de la CoV (une différence importante) des données simulées, la forte volatilité influe beaucoup sur le SCR_{Ultime} et la CoV.

Modèle	***Temps écoulé pour le hyper-paramètrage***	***Temps écoulé pour l'apprentissage***
GLM ODP / Chain Ladder / Mack	-	instantané
Decision Tree	3 min 24 sec	40 msec
Random Forest	36 min 51 sec	295 msec
XGBoost	1 h 1 min	388 msec
MDN	7 h 38 min	14 min 36 msec

TABLE 4.16 : Comparaison temporelle des modèles pour l'environnement 4.

Pour un environnement très volatile, l'analyse coût-bénéfice permet de tirer la conclusion suivante : Si on veut gagner du temps, on va avoir un résultat moins précis avec GLM ODP. Par contre, si on utilise le résultat de XGBoost, on va optimiser le couple coût-bénéfice, mais on ne va pas avoir la densité pour calculer la CoV et le SCR. Dans ce cas, il est préférable d'avoir des résultats plus précis au détriment du temps plutôt qu'une précision inférieure. Pour cet environnement, l'utilisation du MDN est privilégiée.

4.2 Résultats des données réelles

4.2.1 Automobile (RC matériel)

La garantie de responsabilité civile (RC) utilisée couvre le *dommage matériel* plafonné à 100M€.

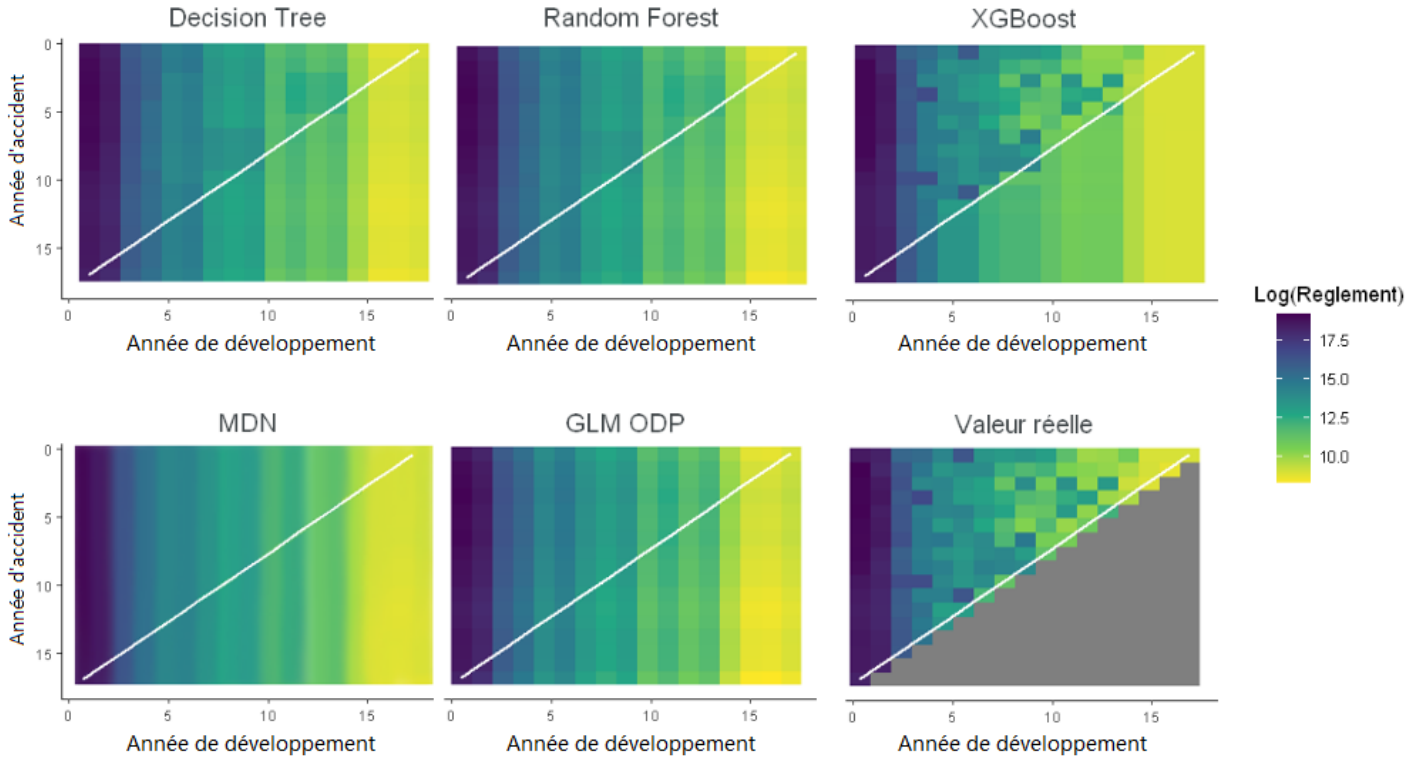

FIGURE 4.17 : Visualisations des réglements prédits pour la garantie automobile (RC matériel).

Les graphiques ci-dessus représentent le log(Réglement prédit) en fonction de l'année d'accident et de l'année de développement. Chaque modèle détecte et projette l'interaction différemment. On observe la nature en bloc de Decision Tree qui segmente le triangle en parties homogènes. Les prédictions des modèles Random Forest et XGBoost semblent lisses puisqu'ils sont fonction d'un certain nombre d'arbres. La prédiction du MDN est la plus lisse et homogène. On observe aussi l'effet des coefficients de développement du GLM ODP.

Comparons le score RMSE d'apprantissage du triangle supérieur de chaque modèle :

Modèle	Score RMSE (M€)
XGBoost	0.74
MDN	1.06
Decision Tree	1.63
Random Forest	1.64
GLM ODP	1.76

TABLE 4.17 : Score RMSE des modèles pour la garantie automobile (RC matériel).

Le XGBoost donne le meilleur score RMSE, avec un écart relatif de 43.24% par rapport au modèle MDN et un écart relatif de 120.27% et de 121.62% par rapport aux modèles Decision Tree et Random Forest respectivement. Il s'éloigne beaucoup du score de GLM ODP avec un écart relatif de 137.83%.

Par la suite, nous examinerons les estimations des provisions des différents modèles en comparant ces derniers avec les provisions calculées par l'expert. D'après la figure (***5.18***), on observe que tous les modèles suivent la tendance des provisions de l'expert. Les provisions calculées par l'expert comprises entre l'année d'accident 2005 et 2014 sont supérieures aux provisions de nos modèles. Cela est expliqué par les jugements d'experts, notamment l'utilisation des choix de coefficients de développement ou de facteurs de queue. On remarque aussi que les modèles de Machine Learning et de Deep Learning résultent des provisions plus importantes pour la dernière année d'accident. En revanche, le GLM ODP se rapproche du niveau de provisions de l'expert pour la dernière année d'accident.

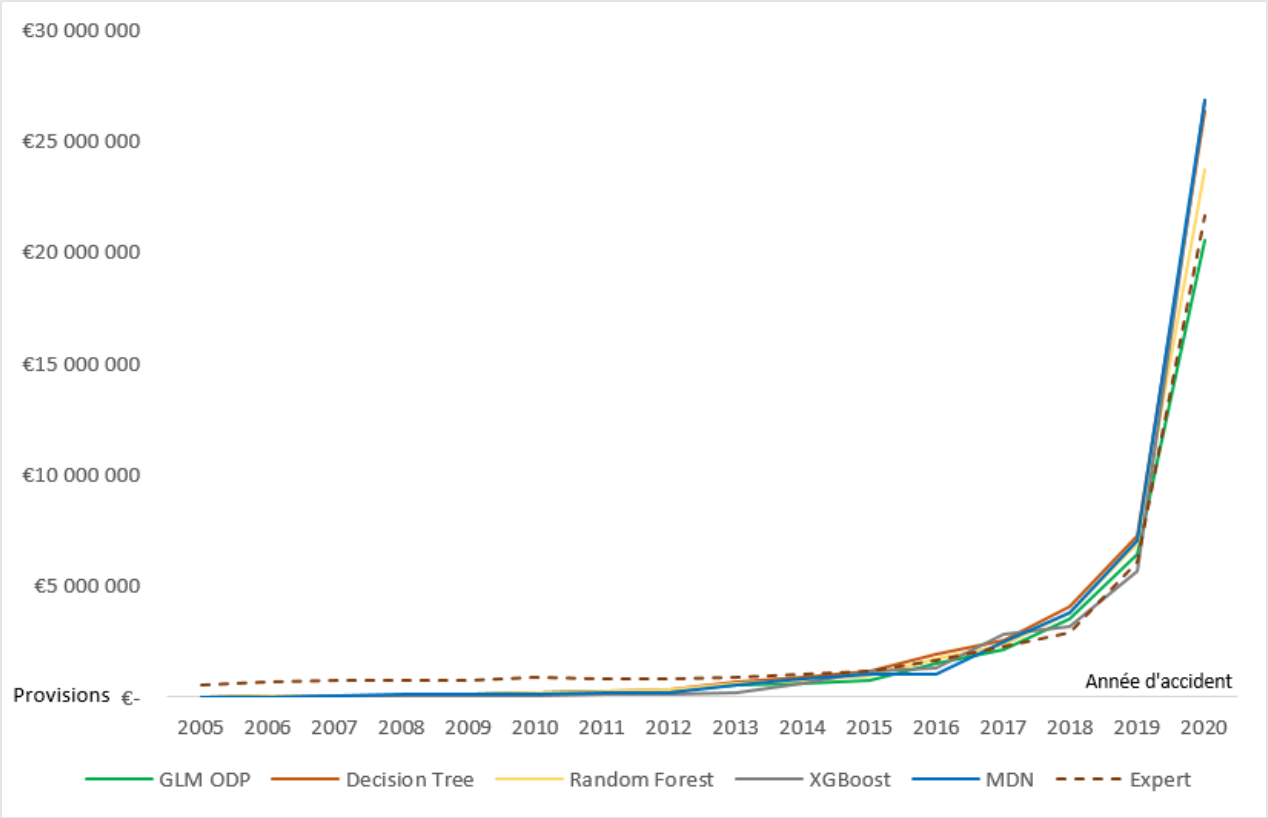

FIGURE 4.18 : Estimations des provisions totales en fonction des années d'accident pour la garantie automobile (RC matériel).

Le tableau (***5.18***) compare les provisions totales de chaque modèle par rapport au calcul d'expert en utilisant un Back-Testing.

Modèle	Provisions (M€)	Ratio (%)
Expert	43.35	100.00
MDN	44.08	102.15
Decision Tree	45.62	105.24
XGBoost	41.98	97.08
Random Forest	41.63	96.76
GLM ODP	36.84	85.13

TABLE 4.18 : Provisions totales et Backtesting pour la garantie automobile (RC matériel).

Les modèles MDN et Decision Tree couvrent respectivement 102.15% et 105.24% des provisions totales calculées par l'expert. Les modèles Random Forest et XGBoost ont des résultats très proches des provisions totales calculées par l'expert, tandis que le GLM ODP couvre moins de provisions totales (85.13%).

Ensuite, nous comparerons les densités des provisions totales, le SCR_{Ultime} et le CoV du GLM ODP et du MDN par rapport à ceux de l'expert.

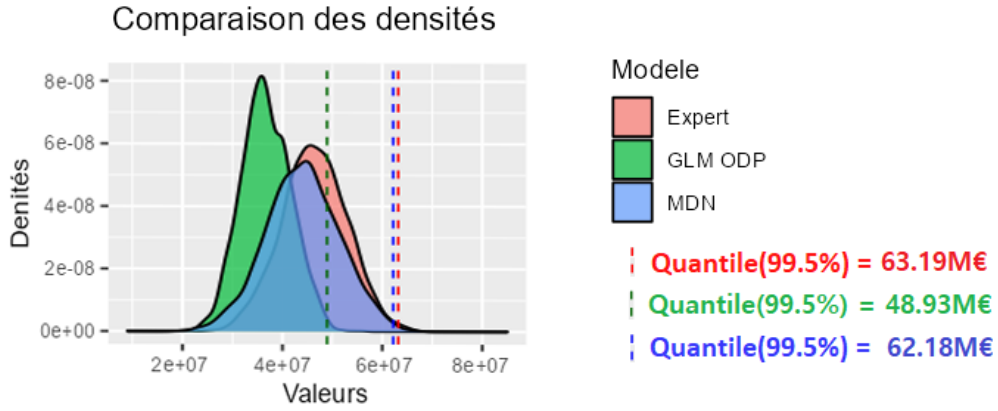

FIGURE 4.19 : Comparaison des densités des provisions totales pour la garantie automobile (RC matériel).

Les densités du MDN et GLM ODP sont très proches de la densité de l'expert. La précision de la distribution du MDN est due à sa capacité à modéliser des tendances complexes dans les données, ainsi qu'à sa capacité à ajuster une distribution plus flexible. La distribution du GLM ODP est obtenue par la méthode Boostrap présentée dans la section ***1.6.3***. La densité de l'expert est obtenue en appliquant un boostrap au modèle Mack présenté dans la section ***1.6.3***.

Modèle	SCR_{Ultime} (M€)	CoV (%)
Expert	19.84	14.08
MDN	18.10	16.67
GLM ODP	12.09	13.54

TABLE 4.19 : SCR_{Ultime} et CoV pour la garantie automobile (RC matériel).

Pour cette garantie, le modèle MDN est plus proche en termes de SCR_{Ultime}, mais sur-estime le CoV en restant proche du résultat de l'expert. En revanche, le SCR_{Ultime} du modèle GLM ODP est sous-estimé, mais le CoV est plus proche que celui du modèle MDN.

Modèle	*Temps écoulé pour le hyper-paramètrage*	*Temps écoulé pour l'apprentissage*
GLM ODP / Chain Ladder / Mack	-	instantané
Decision Tree	1 min 35 sec	31 msec
Random Forest	3 min 46 sec	42 msec
XGBoost	25 min	58 msec
MDN	4 h 12 min	6 min 45 sec

TABLE 4.20 : Comparaison temporelle des modèles pour la garantie automobile (RC matériel).

Pour cette garantie, l'analyse coût-bénéfice permet de tirer la conclusion suivante : Si on veut gagner du temps, on va avoir un résultat moins précis avec GLM ODP. Par contre, si on utilise le modèle Decision Tree, on va optimiser le couple coût-bénéfice, mais on ne va pas avoir la densité des provisions pour calculer la CoV et le SCR. Dans ce cas, il est préférable d'avoir des résultats plus précis au détriment du temps plutôt qu'une précision inférieure. Pour cette garantie, l'utilisation du MDN est privilégié.

4.2.2 Automobile (dommages)

La garantie automobile (dommages) utilisée couvre le *dommage matériel*. Les biens garantis sont indemnisés selon le principe défini par l'article ***L121-1*** du ***code des assurances***.

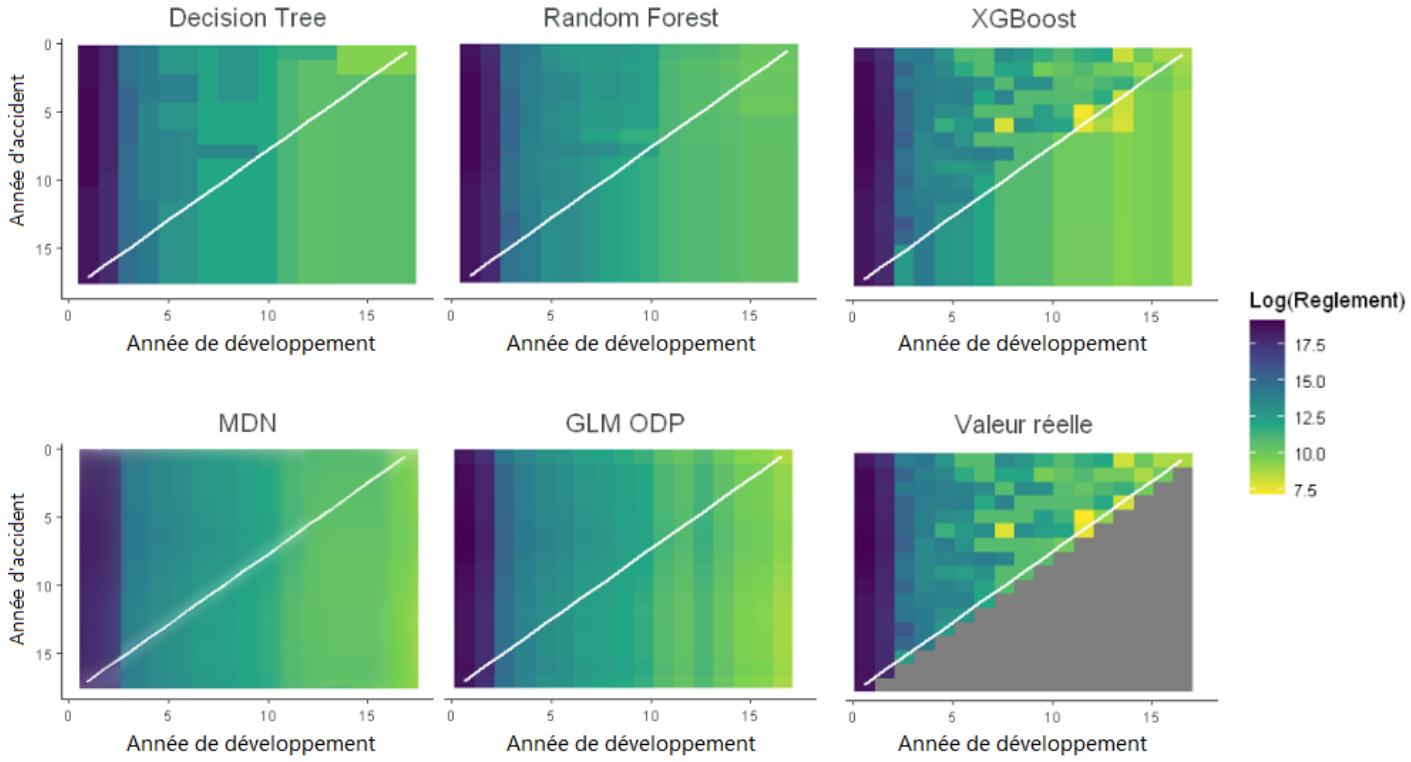

FIGURE 4.20 : Visualisations des réglements prédits pour la garantie automobile (dommages).

Les graphiques ci-dessus montrent que chaque modèle détecte et projette l'interaction différemment. On observe la nature en bloc de Decision Tree qui segmente le triangle en parties homogènes. Les prédictions des modèles Random Forest et XGBoost semblent lisses puisqu'ils sont fonction d'un certain nombre d'arbres. La prédiction du MDN est la plus lisse et homogène. On observe aussi l'effet des coefficients de développement du GLM ODP.

Comparons le score RMSE d'apprantissage du triangle supérieur de chaque modèle :

Modèle	Score RMSE (M€)
XGBoost	1,65
MDN	1,73
Decision Tree	2,12
Random Forest	3,36
GLM ODP	4,67

TABLE 4.21 : Score RMSE des modèles pour la garantie automobile (dommages).

Le XGBoost donne le meilleur score RMSE, avec un écart relatif de 4.84% par rapport au modèle MDN et un écart relatif de 28.48% et de 103.63% par rapport aux modèles Decision Tree et Random Forest respectivement. Il s'éloigne considérablement de celui du GLM ODP avec un écart relatif de 183.03%.

Par la suite, nous examinerons les estimations des provisions des différents modèles en comparant ces derniers avec les provisions calculées par l'expert.

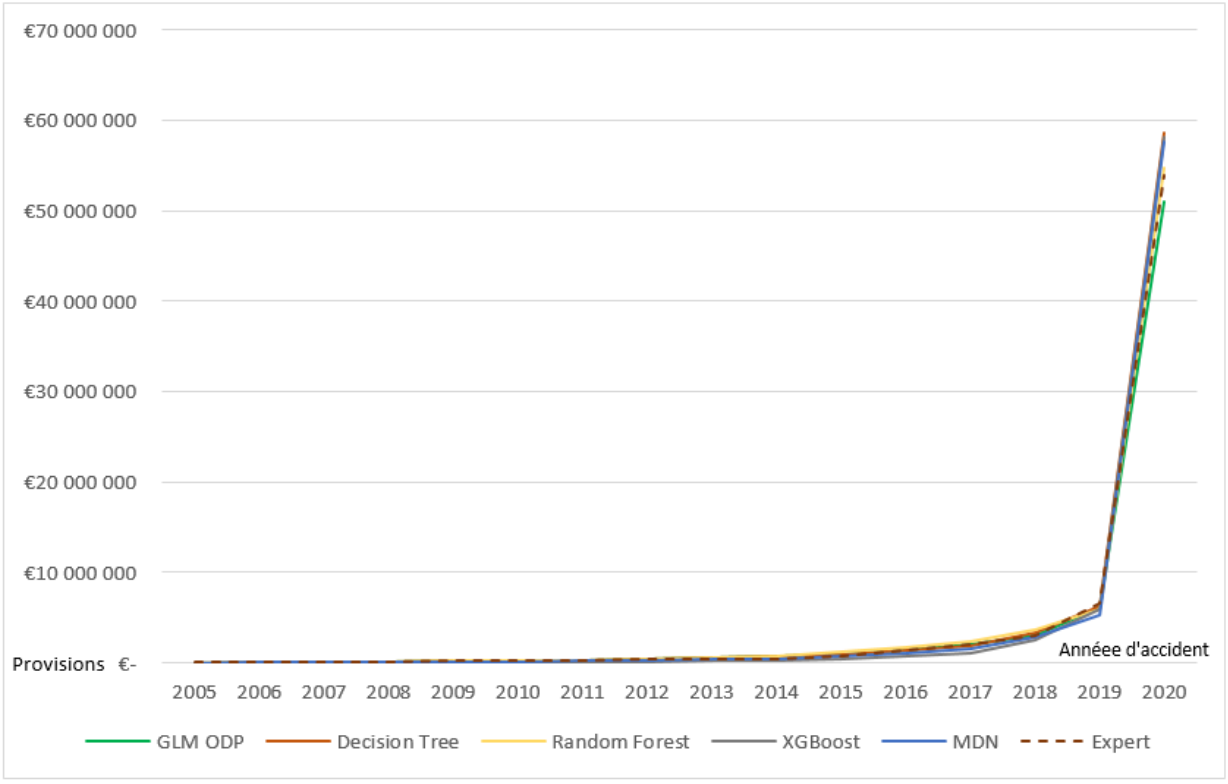

FIGURE 4.21 : Estimations des provisions totales en fonction des années d'accident pour la garantie automobile (dommages).

D'après la figure (***5.21***), on observe que tous les modèles suivent parfaitement la tendance des provisions calculées par l'expert. Il n'y a pas de volatilité remarquable autour de la valeur simulée. On peut affirmer que le modèle GLM ODP, les méthodes de Machine Learning (Decision tree, Random Forest, XGBoost) et les méthodes de Deep Learning (MDN) ont tous réussi à prédire parfaitement l'évolution des provisions totales en fonction d'année d'accident.

Le tableau (***5.22***) compare les provisions totales de chaque modèle par rapport au calcul d'expert en utilisant un Back-Testing.

Modèle	Provisions (M€)	Ratio (%)
Expert	70.28	100.00
MDN	71.20	99.88
XGBoost	70.07	99.70
Random Forest	72.39	103.01
Decision Tree	75.61	107.59
GLM ODP	67.11	95.49

TABLE 4.22 : Provisions totales et Backtesting pour la garantie automobile (dommages).

Les modèles MDN et XGBoost couvrent respectivement 99.88% et 99.70% des provisions totales calculées par l'expert. Les modèles Random Forest et Decision Tree ont des résultats sur-estiment les provisions mais les deux modèles restent proches du résultats calculées par l'expert. Le modèle GLM ODP couvre 95.49.13% de provisions totales ce qui est appréciable.

Ensuite, nous comparerons les densités des provisions totales, le SCR_{Ultime} et le CoV du GLM ODP et du MDN par rapport à ceux de l'expert.

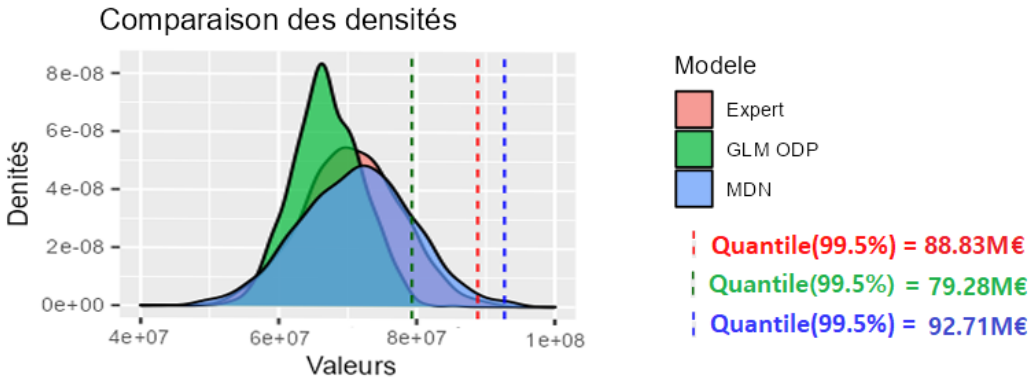

FIGURE 4.22 : Comparaison des densités des provisions totales pour la garantie automobile (dommages).

Les densités du MDN et GLM ODP sont très proches de celle de l'expert. La précision de la distribution du MDN est due à sa capacité à modéliser des tendances complexes dans les données, ainsi qu'à sa capacité à ajuster une distribution plus flexible (modèle de mélange gaussien). La distribution du GLM ODP obtenue par la méthode Boostrap présentée dans la section ***1.6.3*** donne aussi un résultat très satisfaisant pour cette garantie. La densité de l'expert est obtenue en appliquant un boostrap au modèle Mack présentée dans la section ***1.6.3***.

D'après le tableau (***5.23***), le MDN est plus précis en termes de SCR_{Ultime} et CoV. En revanche, le SCR_{Ultime} et le CoV du GLM ODP sont aussi proches mais avec une légère sous-estimation.

Modèle	SCR_{Ultime} (M€)	CoV (%)
Expert	18.24	10.65
MDN	21.44	11.38
GLM ODP	12.23	7.52

TABLE 4.23 : SCR_{Ultime} et CoV pour la garantie automobile (dommages).

Les méthodes de Machine Learning et de Deep Learning sont plus précises que les méthodes usuelles. Néanmoins, d'après le tableau (***5.24***), leurs traitements sont chronophages et énergivores (pour le hyper-paramètres et l'apprentissage) relativement aux méthodes usuelles, qui donnent les résultats instantanément.

Modèle	*Temps écoulé pour le hyper-paramètrage*	*Temps écoulé pour l'apprentissage*
GLM ODP / Chain Ladder / Mack	-	instantané
Decision Tree	1 min 56 sec	39 msec
Random Forest	3 min 58 sec	47 msec
XGBoost	24 min	11 msec
MDN	4 h 06 min	6 min 32 sec

TABLE 4.24 : Comparaison temporelle des modèles pour la garantie automobile (dommages).

Pour cette garantie, l'analyse coût-bénéfice permet de tirer la conclusion suivante : il vaut mieux utiliser les méthodes usuelles que celles de Machine Learning ou Deep Learning, pour garner beaucoup de temps, en sacrifiant une légère précision.

4.2.3 Responsabilité civile générale

La garantie responsabilité civile générale (RCG) utilisée couvre le dommage matériel causé par une personne tenu responsable d'avoir causé involontairement ce dommage matériel à un tiers, soit par un produit ou par des activités d'exploitation.

Les graphiques ci-dessous montrent que chaque modèle détecte et projette l'interaction différemment. On obsèrve la nature en bloc de Decision Tree qui segmente le triangle en parties homogènes. Les prédictions des modèles Random Forest et XGBoost semblent lisses puisqu'ils sont fonction d'un certain nombre d'arbres. La prédiction du MDN est la plus lisse et homogène. On observe aussi l'effet des coefficients de développement du GLM ODP.

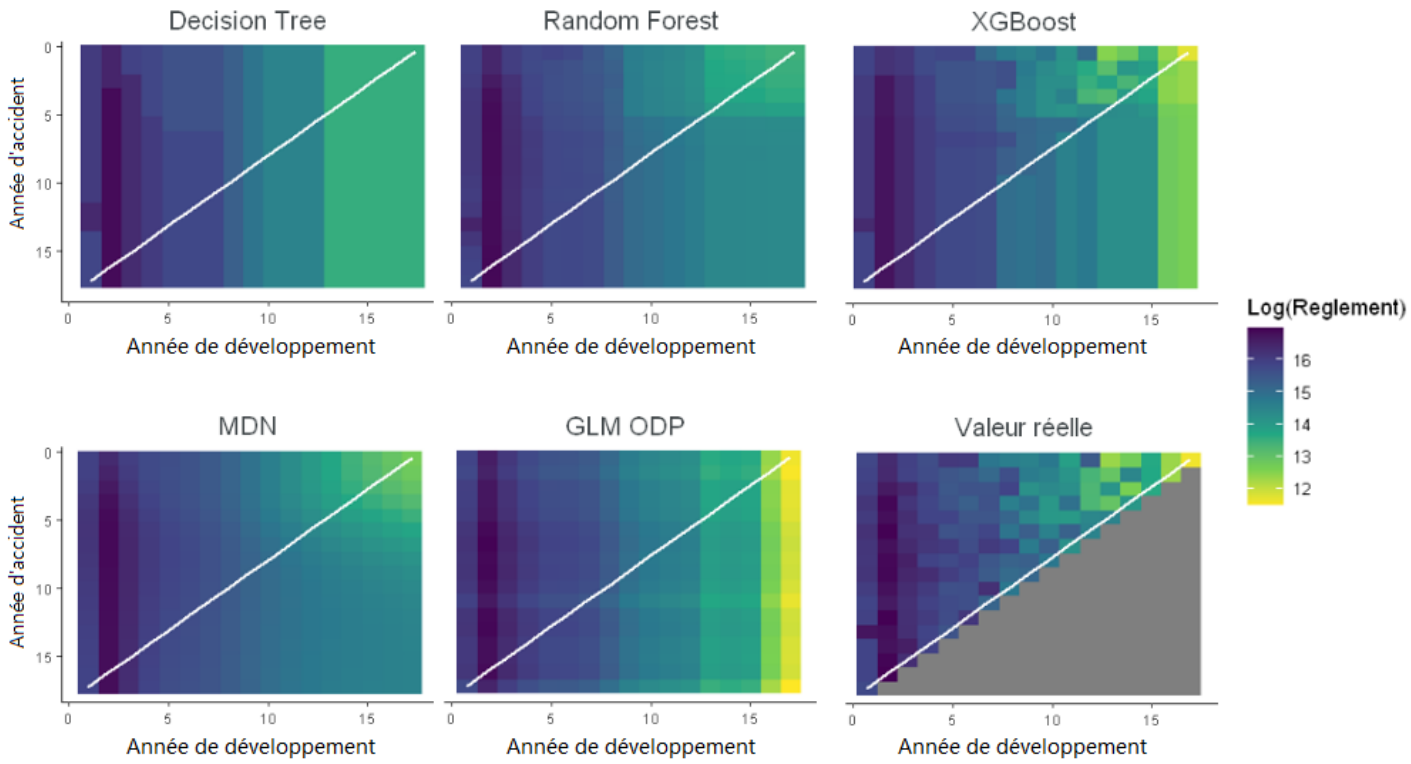

FIGURE 4.23 : Visualisations des réglements prédits pour la garantie responsabilité civile générale.

Comparons le score RMSE d'apprantissage du triangle supérieur de chaque modèle :

Modèle	Score RMSE (M€)
MDN	1, 33
XGBoost	1, 46
Random Forest	1, 56
Decision Tree	2, 73
GLM ODP	3, 63

TABLE 4.25 : Score RMSE des modèles pour la garantie responsabilité civile générale.

Le MDN donne le meilleur score RMSE, avec un écart relatif de 9.77% par rapport au modèle XGBoost et un écart relatif de 17.29% et de 105.26% par rapport aux modèles Random Forest et Decision Tree respectivement. Il s'éloigne considérablement du score de GLM ODP avec un écart relatif de 172.93%.

Par la suite, nous examinerons les estimations des provisions des différents modèles en comparant ces derniers avec les provisions calculés par l'expert. D'après la figure (***5.24***), on observe que tous les modèles suivent la tendance des provisions de l'expert avant 2019. Après 2019, les méthodes de Machine Learning et de Deep Learning ne suivent pas la tendance que le GLM ODP et l'expert ont suivi.Cela est expliqué par les jugements d'experts, notamment l'utilisation des choix de coefficients de développement ou de facteurs de queue. Les provisions calculées par l'expert comprises entre l'année d'accident 2005 et 2012 sont supérieures aux provisions de des modèles utilisés. On remarque aussi que les modèles de Machine Learning et de Deep Learning sur-estiment les provisions pour la dernière année d'accident (2019 et 2020).

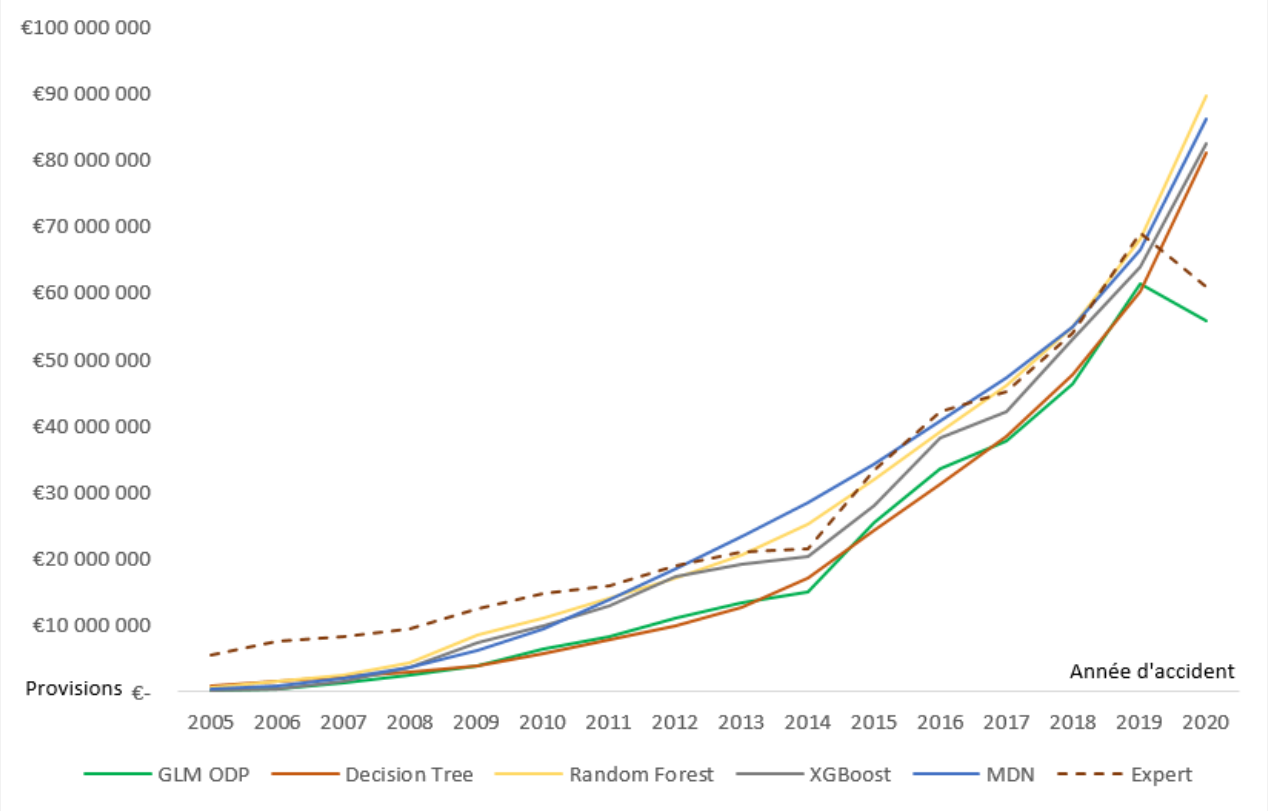

FIGURE 4.24 : Estimations des provisions totales en fonction des années d'accident pour la garantie responsabilité civile générale

Le tableau (***5.26***) compare les provisions totales de chaque modèle par rapport au calcul d'expert en utilisant un Back-Testing.

Modèle	Provisions (M€)	Ratio (%)
Expert	439.64	100.00
MDN	435.28	99.01
Random Forest	434.59	98.85
XGBoost	400.21	91.03
Decision Tree	347.16	78.96
GLM ODP	322.56	73.37

TABLE 4.26 : Provisions totales et Backtesting pour la garantie responsabilité civile générale.

Les modèle MDN, Random Forest et XGBoost couvrent respectivement 99.01%, 98.85% et 91.03% des provisions totales calculées par l'expert, ce qui est appréciable. Les modèles Decision Tree et GLM ODP couvrent mois de provisions, à savoir 78.96% et 73.37% respectivement.

Ensuite, nous comparerons les densités des provisions totales, le SCR_{Ultime} et le CoV du GLM ODP et du MDN par rapport à ceux de l'expert.

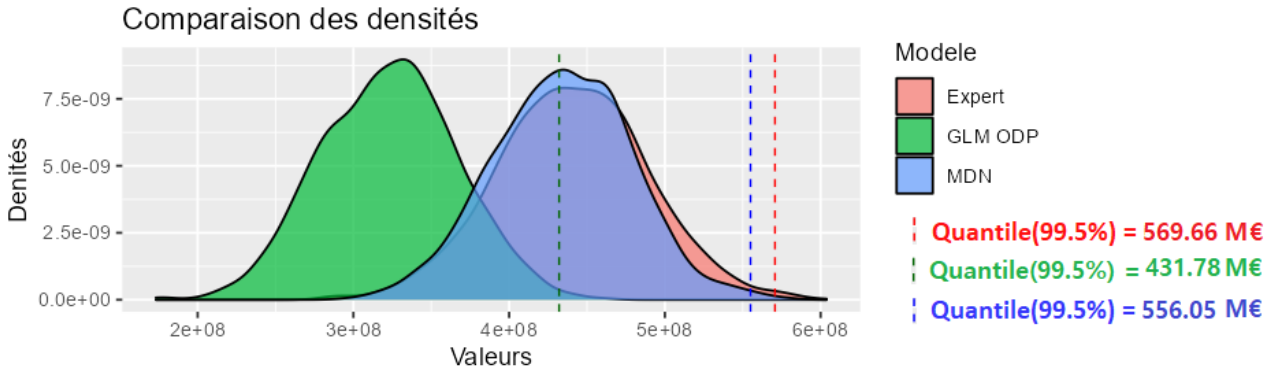

FIGURE 4.25 : Comparaison des densités des provisions totales pour la garantie responsabilité civile générale.

La densité du MDN ne s'éloigne pas de celle de l'expert. Par contre, la densité du GLM ODP s'éloigne un peu de la densité de l'expert. La précision de la distribution du MDN est due à sa capacité à modéliser des tendances complexes dans les données, ainsi qu'à sa capacité à ajuster une distribution plus flexible. La distribution du GLM ODP obtenue par la méthode Boostrap présentée dans la section ***1.6.3***, ne réussi pas à se rapprocher de la densité des données simulées. La densité de l'expert est obtenu en appliquant un boostrap au modèle Mack présenté dans la section ***1.6.3***.

Modèle	SCR_{Ultime} (M€)	CoV (%)
Expert	130.02	10.65
MDN	120.77	10.21
GLM ODP	109.22	11.00

TABLE 4.27 : SCR_{Ultime} et CoV ppour la garantie responsabilité civile générale.

Pour cette garantie, le modèle MDN est plus précis en termes de SCR_{Ultime}, mais sous-estime le CoV en restant proche. En revanche, le SCR_{Ultime} du modèle GLM ODP est sous-estimé, mais le CoV est plus proche que celui du modèle MDN.

Modèle	*Temps écoulé pour le hyper-paramètrage*	*Temps écoulé pour l'apprentissage*
GLM ODP / Chain Ladder / Mack	-	instantané
Decision Tree	1 min 35 sec	31 msec
Random Forest	3 min 46 sec	42 msec
XGBoost	25 min	58 msec
MDN	4 h 12 min	6 min 45 sec

TABLE 4.28 : Comparaison temporelle des modèles pour la garantie responsabilité civile générale.

Pour cette garantie, l'analyse coût-bénéfice permet de tirer la conclusion suivante : Si on veut gagner du temps, on va avoir un résultat moins précis avec GLM ODP. Par contre, si on utilise le modèle Random Forest, on va optimiser le couple coût-bénéfice, mais on ne va pas avoir la densité des provisions pour calculer la CoV et le SCR. Dans ce cas, il est préférable d'avoir des résultats plus précis au détriment du temps plutôt qu'une précision inférieure. Pour cette garantie, l'utilisation du MDN est privilégié.

4.3 Avantages et limites des modèles

La méthode ROCV a permis d'atteindre deux objectifs dans ce mémoire :

1) Partitionner le triangle supérieur en ensembles de formation, de validation et de test. Cela a permis de tester différents modèles et différentes combinaisons d'hyper-paramètres et de les sélectionner en fonction de leur précision dans l'ensemble de test.

2) Évaluer spécifiquement les différents modèles en fonction de leur précision de projection, ce qui augmente les chances de sélectionner un modèle qui se projette avec précision dans le triangle inférieur. Ceci est réalisé par le partitionnement séquentiel des données, de sorte que les données de test sont constituées des derniers trimestres calendaires.

D'ailleurs, l'ensemble des hyper-paramètres sélectionnés par la méthode ROCV a produit des prédictions des provisions totales (et de la distribution des provisions pour le modèle MDN) très raisonnables et précises.

En revanche, la méthode ROCV n'a pas permis de détecter avec une grande précision les tendances intégrées dans les derniers trimestres calendaires. Dans l'environnement 3, les méthodes de Machine Learning et de Deep Learning n'ont pas réussi à détecter avec précision le choc d'inflation pour les derniers trimestres, car ils n'ont pas été entraînés sur beaucoup de données où le choc a pris effet. La première partition, en particulier, est à peine entraînée ou validée sur les derniers trimestres calendaires, qui contiennent des informations cruciales pour la projection.

La robustesse du modèle MDN peut être partiellement attribuée à son ajustement lisse de la moyenne. La méthode ROCV, dans la troisième partition, utilise les derniers trimestres calendaires pour la validation. Un modèle qui surcharge les données sans partitionnement ne peut pas faire de projections précises, c'est pourquoi les méthodes de Machine Learning et de Deep Learning sont favorisées pour produire des ajustements plus lisses et plus robustes grâce à la méthode ROCV.

Les modèles XGBoost et MDN ont obtenu des prédictions des provisions totales plus précises que celles des méthodes usuelles (GLM ODP) dans tous les environnements testés, même dans le cas d'un environnement où les hypothèses de Chaine-Ladder ne sont pas applicables (environnement 4). En outre, l'approche MDN à surpasser l'approche GLM ODP boostrapé au niveau de l'estimation de la distribution des provisions, le SCR_{Ultime} et le CoV.

Par contre, la quantité limitée de données par rapport aux grands ensembles de données auxquels les méthodes de Machine Learning et Deep Learning sont habitués découragerait l'utilisation de la méthode ROCV. Cependant, cette méthode appliquée aux modèles a donné d'excellents résultats pour un triangle de développement 40x40 (pour les données simulées) et 17x17 (pour les données réelles), ce qui rend ces modèles plus appropriés dans un cadre pratique les prochaines années en provisionnement non-vie.

Malgré leurs perfomances, les méthodes de Machine Learning et de Deep Learning rencontrent plusieurs limites, on peur citer quelques uns :

- ▷ Les modèles nécessitent souvent un temps plus important que celui des méthodes usuelles pour avoir les résultats.
- ▷ Le manque d'interprétabilité et la capacité d'expliquer, de piloter ou de présenter des informations dans des termes humainement compréhensibles.

- ▷ Quelques modèles requièrent que les montants incrémentaux $X_{i,j}$ soint positifs, tandis que les montants incrémentaux $X_{i,j}$ sont souvent négatifs à cause de la présence de boni de liquidation dans le déroulement de la charge des sinistres, ou l'encaissement de recours en fin de développement, pour les triangles de règlements.
- ▷ Les échelles de complexité algorithmique des méthodes de Machine Learning et de Deep Learning augmentent la difficulté d'acquérir les compétences acquises pour les appliquer. En effet, ce n'est pas quelque chose que l'on peut acquérir rapidement avec la charge de travail des actuaires concernant leurs projets et leurs missions.

Les modèles de Machine Learning et Deep Learning avec la méthode ROCV ouvrent des perspectives d'automatisation et pourraient, par exemple, être utile dans le cadre de processus de revue indépendante de provisionnement.

Conclusion

Plusieurs obstacles entravant l'application des méthodes de Machine Learning et de Deep Learning sur les triangles de développement dans le domaine de provisionnement non-vie ont été identifiés, abordés et améliorés dans ce mémoire. L'absence de prévision distributive dans le domaine de la gestion des sinistres a été résolue avec succès grâce à un réseau de neurones de densité mélange "MDN". La méthode ROCV fournit un cadre de test et de sélection de modèles en améliorant la précision de la projection et en facilitant la mise en œuvre de la modélisation des méthodes de Machine Learning utilisées (Decision Tree, Random Forest et XGBoost avec une régression tweedie) et du MDN dans la pratique.

Ce mémoire n'a pas pour but de remplacer les méthodes usuelles en provisionnement mais de les concurrencer et d'offrir plus de choix pour l'estimation du BE et de la densité de provisions totales. Les méthodes usuelles avec un pilotage optimal en adaptant les coefficients de développement selon l'expérience et l'historique de la garantie étudiée aboutissent à des résultats très performants et interprétables avec un gain de temps considérable.

La flexibilité de la distribution gaussienne mélange a permis au MDN de consacrer sa capacité de modélisation à l'obtention des prédictions du BE et de la distribution avec une précision remarquable. La régression tweedie appliquée au modèle XGBoost a amélioré les prédictions de provisions totales. Dans notre étude, les méthodes de Machine Learning et Deep Learning ont dépassé les méthodes usuelles en termes de précision.

Ces résultats ont été obtenus dans quatre environnements différents, avec des complexités structurelles variables. Le XGBoost et le MDN ont réussi à capturer partiellement la variation de la cadence des sinistres, le choc d'inflation. Le MDN a fourni des prévisions distributionnelles stables et précises avec un ensemble de données très volatiles. Ces résultats montrent que les modèles MDN et XGBoost peuvent être appliqués avec succès à une grande variété de triangles de développement de différentes branches d'activité avec des complexités liées à l'entreprise.

Ce mémoire effectue une modélisation sur les triangles de développement uniquement. La modélisation des sinistres individuels a pris de l'ampleur depuis que les méthodes de Machine Learning et de Deep Learning ont été popularisées. La modélisation des sinistres individuels analyse les sinistres à un niveau plus détaillé, ce qui permet d'utiliser plus efficacement la puissance de modélisation des méthodes de Machine Learning et de Deep Learning. Notre étude s'est focalisée sur l'utilisation du mélange gaussien pour tester les performances du modèle MDN. Le fait de permettre au MDN d'utiliser une plus grande variété de densités de composants, comme l'exponentiel, gamma ou pareto, augmente sa polyvalence.

D'ailleurs, l'utilisation des modèles XGBoost et MDN nécessite un temps important pour calibrer les hyper-paramètres et pour la prédiction, cela est considéré comme un obstacle principal pour les actuaires et les data scientist. Ce problème pourra sûrement être résolu dans les prochaines années avec le développement des ordinateurs quantiques.

Bibliographie

Avanzi, B. et Taylor (2020). SynthETIC: an individual insurance claim simulator with feature control.

Balona, C. et Richman, R. (2020). The Actuary and IBNR Techniques: a Machine Learning Approach. *SSRN 3697256*, p. 1-53.

Bergmeir, C. et Bentez (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences* 191.2, p. 192-213.

Bishop, C. M. (1994). Mixture Density Networks. *Technical Report. Aston University, Birmingham.*

Bornhuetter, R. et Ferguson, R. (1972). The Actuary and IBNR. *Proceedings of the Casualty Actuarial Society Casualty* 1.1, p. 181-195.

Breiman, L (1996). Bagging Predictors. *Machine Learning* 24, 123-–140.

Breiman, L (2001). Random Forests. *Machine Learning* 45, 5—32.

Cun, L. (1986). A Theoretical Framework for Back-Propagation. *Proceedings of the 1988*, 21–28.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* 2, 303-–314.

Cybenko, G. (2020). Collective reserving using individual claims data. *Scandinavian Actuarial Journal* 1, 1-–28.

Friedland, J. (2020). Survey of Canadian Actuaries on ML in Reserving. Article.

Friedman, J. H. (1999). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* 29, p. 1189-1232.

Gabrielli, A. (2019). A neural network boosted double overdispersed poisson claims reserving model. *ASTIN Bulletin: The Journal of the IAA* 50.2, p. 25-60.

Gabrielli, A. (2020a). An individual claims reserving model for reported claims. *European Actuarial Journal* 11, 541—577.

Gabrielli, A. (2020b). Neural network embedding of the over-dispersed poisson reserving model. *Scandinavian Actuarial Journal* 2020.1, p. 1-29.

Guillaumes, A. (2017). Mixture Density Networks for distribution and uncertainty estimation. *Mémoire du Master Artificial Intelligence, Université de Barcelone.*

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks* 4.2, p. 251-257.

Kuo, K. (2019). DeepTriangle: A deep learning approach to loss reserving. *Risks* 7.3, p. 97.

Kuo, K. (2020). Individual Claims Forecasting with Bayesian Mixture Density Networks. *arXiv preprint arXiv* 1.1, p. 24-53.

M. Taher Al-Mudafer B. Avanzi, G. T. B. W. (2021). Stochastic loss reserving with mixture density neural networks. *Insurance: Mathematics and Economics* 105, p. 144-174.

MacDonnell, S. (2020). Machine learning in Reserving Working Party - UK survey findings. Rapport.

Mack, T. (1993). Distribution-free formula for the standard error of chain ladder reserve estimates. *ASTIN Bulletin: The Journal of the IAA* 29.2, p. 361-366.

Mario Wuthrich, V. (2018). Individual Claims History Simulation Machine. *Risks* 101.1, p. 80-92.

Mulquiney, P. (2006). Artificial Neural Networks in Insurance Loss Reserving. *Proceedings of the 9th Joint International Conference on Information Sciences (JCIS-06)*. Atlantis Press.

Ormoneit et Tresp (1996). Improved Gaussian Mixture Density Estimates Using Bayesian Penalty Terms and Network Averaging. *Advances in Neural Information Processing Systems* 1.1, p. 1-548.

Renshaw, A. et Verrall, R. (1998). A Stochastic Model Underlying the Chain-Ladder Technique. *British Actuarial Journal* 4.4, p. 903-923.

Rossouw, L. et Richman, R. (2019). Using Machine Learning to Model Claims Experience and Reporting Delays for Pricing and Reserving. *SSRN 3465424*, p. 1-44.

Scarth, R. (2020). A Practitioner's Introduction to Stochastic Reserving - The One-Year View. Article.

Wikistat (2016). Réseaux de neurones — Wikistat. url : http://wikistat.fr/pdf/st-m-app-rn.pdf.

Wüthrich, M. V. (2018). Neural Networks Applied to Chain–Ladder Reserving. *European Actuarial Journal* 8.2, p. 407-436.

Wuthrich, M. (2019). From generalized linear models to neural networks, and background. *RiskLab* 1.1, p. 903-923.

Yang, L. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415, p. 295-316.

Zen, H. (2014). Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. *014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Annexe A

Compléments

A.1 Les hyper-paramètres utilisés

A.1.1 Les hyper-paramètres de Decision Tree (Arbre de décision)

Les hyperparamètres sont des paramètres réglables qui permettent de contrôler le processus d'apprentissage du modèle pour donner une meilleur prédiction.

Hyper-paramètres	Plage de valeurs	Valeur optimale
cp	$[10^{-4}, 1]$	10^{-4}
minsplit	$\{1, 2, ..., 60\}$	20
maxdepth	$\{1, 2, ..., 30\}$	27

TABLE A.1 : Hyper-paramètres de Decision Tree pour l'environnement 1.

Hyper-paramètres	Plage de valeurs	Valeur optimale
cp	$[10^{-4}, 1]$	10^{-4}
minsplit	$\{1, 2, ..., 60\}$	20
maxdepth	$\{1, 2, ..., 30\}$	27

TABLE A.2 : Hyper-paramètres de Decision Tree pour l'environnement 2.

Hyper-paramètres	Plage de valeurs	Valeur optimale
cp	$[10^{-6}, 1]$	10^{-6}
minsplit	$\{1, 2, ..., 30\}$	24
maxdepth	$\{1, 2, ..., 30\}$	27

TABLE A.3 : Hyper-paramètres de Decision Tree pour l'environnement 3.

Hyper-paramètres	Plage de valeurs	Valeur optimale
cp	$[10^{-4}, 1]$	10^{-4}
minsplit	$\{1, 2, ..., 30\}$	24
maxdepth	$\{1, 2, ..., 30\}$	4

TABLE A.4 : Hyper-paramètres de Decision Tree pour l'environnement 4.

Hyper-paramètres	Plage de valeurs	Valeur optimale
cp	$[0, 1]$	0
minsplit	$\{1, 2, ..., 30\}$	10
maxdepth	$\{1, 2, ..., 30\}$	30

TABLE A.5 : Hyper-paramètres de Decision Tree pour la garantie automobile (responsabilité civile).

Hyper-paramètres	Plage de valeurs	Valeur optimale
cp	$[0, 1]$	0
minsplit	$\{1, 2, ..., 30\}$	10
maxdepth	$\{1, 2, ..., 30\}$	26

TABLE A.6 : Hyper-paramètres de Decision Tree pour la garantie automobile (dommages).

Hyper-paramètres	Plage de valeurs	Valeur optimale
cp	$[0, 1]$	0
minsplit	$\{1, 2, ..., 30\}$	9
maxdepth	$\{1, 2, ..., 30\}$	4

TABLE A.7 : Hyper-paramètres de Decision Tree pour la garantie responsabilité civile générale.

A.1.2 Les hyper-paramètres de Random Forest (Forêt aléatoire)

Hyper-paramètres	Plage de valeurs	Valeur optimale
ntree	$\{1, 2, ..., 1000\}$	112
mtry	$\{1, 2\}$	2
nodesize	$\{1, 2, ..., 40\}$	18
maxnodes	$\{1, 2, ..., 40\}$	36

TABLE A.8 : Hyper-paramètres de Random Forest pour l'environnement 1.

Hyper-paramètres	Plage de valeurs	Valeur optimale
ntree	$\{1, 2, ..., 1500\}$	112
mtry	$\{1, 2\}$	2
nodesize	$\{1, 2, ..., 60\}$	7
maxnodes	$\{1, 2, ..., 60\}$	60

TABLE A.9 : Hyper-paramètres de Random Forest pour l'environnement 2.

Hyper-paramètres	Plage de valeurs	Valeur optimale
ntree	$\{1, 2, ..., 1000\}$	984
mtry	$\{1, 2\}$	2
nodesize	$\{1, 2, ..., 40\}$	28
maxnodes	$\{1, 2, ..., 40\}$	38

TABLE A.10 : Hyper-paramètres de Random Forest pour l'environnement 3.

Hyper-paramètres	Plage de valeurs	Valeur optimale
ntree	$\{1, 2, ..., 1000\}$	334
mtry	$\{1, 2\}$	2
nodesize	$\{1, 2, ..., 40\}$	18
maxnodes	$\{1, 2, ..., 40\}$	18

TABLE A.11 : Hyper-paramètres de Random Forest pour l'environnement 4.

Hyper-paramètres	Plage de valeurs	Valeur optimale
ntree	$\{1, 2, ..., 1000\}$	112
mtry	$\{1, 2\}$	2
nodesize	$\{1, 2, ..., 40\}$	20
maxnodes	$\{1, 2, ..., 40\}$	30

TABLE A.12 : Hyper-paramètres de Random Forest pour la garantie automobile (responsabilité civile).

Hyper-paramètres	Plage de valeurs	Valeur optimale
ntree	$\{1, 2, ..., 1000\}$	667
mtry	$\{1, 2\}$	2
nodesize	$\{1, 2, ..., 40\}$	1
maxnodes	$\{1, 2, ..., 40\}$	40

TABLE A.13 : Hyper-paramètres de Random Forest pour la garantie automobile (dommages).

Hyper-paramètres	Plage de valeurs	Valeur optimale
ntree	$\{1, 2, ..., 1000\}$	334
mtry	$\{1, 2\}$	2
nodesize	$\{1, 2, ..., 40\}$	1
maxnodes	$\{1, 2, ..., 40\}$	23

TABLE A.14 : Hyper-paramètres de Random Forest pour la garantie responsabilité civile générale.

A.1.3 Les hyper-paramètres de XGBoost

Hyper-paramètres	Plage de valeurs	Valeur optimale
objective	–	$reg : tweedie$
tweedie_variance_power	$[1.01, 1.99]$	1.87
eta	$[0.001, 0.8]$	0.27
gamma	$[0.0, 0.5]$	0.28
max_depth	$\{2, ..., 500\}$	244
nrounds	$\{100, .., 600\}$	152
lambda	$[0.01, 0.8]$	0.09

TABLE A.15 : Hyper-paramètres de Random Forest pour l'environnement 1.

Hyper-paramètres	Plage de valeurs	Valeur optimale
objective	–	$reg : tweedie$
tweedie_variance_power	$[1.01, 1.99]$	1.91
eta	$[0.001, 0.8]$	0.21
gamma	$[0.0, 0.5]$	0.34
max_depth	$\{2, ..., 500\}$	324
nrounds	$\{100, .., 600\}$	409
lambda	$[0.01, 0.8]$	0.75

TABLE A.16 : Hyper-paramètres de Random Forest pour l'environnement 2.

Hyper-paramètres	Plage de valeurs	Valeur optimale
objective	–	$reg : tweedie$
tweedie_variance_power	$[1.01, 1.99]$	1.58
eta	$[0.001, 0.8]$	0.09
gamma	$[0.0, 0.5]$	0.04
max_depth	$\{2, ..., 500\}$	2
nrounds	$\{100, .., 600\}$	287
lambda	$[0.01, 0.8]$	0.50

TABLE A.17 : Hyper-paramètres de Random Forest pour l'environnement 3.

Hyper-paramètres	Plage de valeurs	Valeur optimale
objective	–	$reg : tweedie$
tweedie_variance_power	$[1.01, 1.99]$	1.96
eta	$[0.001, 0.5]$	0.19
gamma	$[0.0, 30]$	13
max_depth	$\{2, ..., 800\}$	790
nrounds	$\{100, .., 800\}$	155
lambda	$[0.01, 0.8]$	0.36
subsample	$[0.01, 0.8]$	0.77
min_child_weight	$\{1, 2, ..., 5\}$	4
colsample_bytree	$[0.01, 0.8]$	0.62

TABLE A.18 : Hyper-paramètres de Random Forest pour l'environnement 4.

Hyper-paramètres	Plage de valeurs	Valeur optimale
objective	–	$reg : tweedie$
tweedie_variance_power	$[1.01, 1.99]$	1.81
eta	$[0.001, 0.8]$	0.32
gamma	$[0.0, 0.5]$	0.47
max_depth	$\{2, ..., 500\}$	70
nrounds	$\{100, .., 600\}$	547
lambda	$[0.01, 0.8]$	0.14

TABLE A.19 : Hyper-paramètres de Random Forest pour la garantie automobile (responsabilité civile).

Hyper-paramètres	Plage de valeurs	Valeur optimale
objective	–	$reg : tweedie$
tweedie_variance_power	$[1.01, 1.99]$	1.75
eta	$[0.001, 0.8]$	0.72
gamma	$[0.0, 0.5]$	0.35
max_depth	$\{2, ..., 500\}$	662
nrounds	$\{100, .., 600\}$	261
lambda	$[0.01, 0.8]$	0.71

TABLE A.20 : Hyper-paramètres de Random Forest pour la garantie automobile (dommages).

Hyper-paramètres	Plage de valeurs	Valeur optimale
objective	–	$reg : tweedie$
tweedie_variance_power	$[1.01, 1.99]$	1.94
eta	$[0.001, 0.8]$	0.58
gamma	$[0.0, 0.5]$	0.48
max_depth	$\{2, ..., 500\}$	3
nrounds	$\{100, .., 600\}$	488
lambda	$[0.01, 0.8]$	0.12

TABLE A.21 : Hyper-paramètres de Random Forest pour la garantie responsabilité civile générale.

A.1.4 Les hyper-paramètres de MDN

Hyper-paramètres	Plage de valeurs	Valeur optimale
λ_w	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	0
λ_σ	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	10^{-1}
dropout (p)	$\{0, 0.1, 0.2\}$	0
neurones (n)	$\{20, 40, 60, 80, 100\}$	80
hidden (h)	$\{1, 2, 3, 4, 5\}$	4
composents (K)	$\{1, 2, 3, 4\}$	3

TABLE A.22 : Hyper-paramètres de MDN pour l'environnement 1.

Hyper-paramètres	Plage de valeurs	Valeur optimale
λ_w	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	0
λ_σ	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	10^{-4}
dropout (p)	$\{0, 0.1, 0.2\}$	0.2
neurones (n)	$\{20, 40, 60, 80, 100\}$	80
hidden (h)	$\{1, 2, 3, 4, 5\}$	2
composents (K)	$\{1, 2, 3, 4\}$	2

TABLE A.23 : Hyper-paramètres de MDN pour l'environnement 2.

Hyper-paramètres	Plage de valeurs	Valeur optimale
λ_w	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	10^{-3}
λ_σ	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	10^{-2}
dropout (p)	$\{0, 0.1, 0.2\}$	0
neurones (n)	$\{20, 40, 60, 80, 100\}$	20
hidden (h)	$\{1, 2, 3, 4, 5\}$	2
composents (K)	$\{1, 2, 3, 4\}$	1

TABLE A.24 : Hyper-paramètres de MDN pour l'environnement 3.

Hyper-paramètres	Plage de valeurs	Valeur optimale
λ_w	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	0
λ_σ	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	10^{-2}
dropout (p)	$\{0, 0.1, 0.2\}$	0
neurones (n)	$\{20, 40, 60, 80, 100\}$	80
hidden (h)	$\{1, 2, 3, 4, 5\}$	4
composents (K)	$\{1, 2, 3, 4\}$	3

TABLE A.25 : Hyper-paramètres de MDN pour l'environnement 4.

Hyper-paramètres	Plage de valeurs	Valeur optimale
λ_w	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	10^{-2}
λ_σ	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	10^{-1}
dropout (p)	$\{0, 0.1, 0.2\}$	0.2
neurones (n)	$\{20, 40, 60, 80, 100\}$	100
hidden (h)	$\{1, 2, 3, 4, 5\}$	3
composents (K)	$\{1, 2, 3, 4\}$	2

TABLE A.26 : Hyper-paramètres de MDN pour la garantie automobile (responsabilité civile).

Hyper-paramètres	Plage de valeurs	Valeur optimale
λ_w	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	10^{-2}
λ_σ	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	10^{-1}
dropout (p)	$\{0, 0.1, 0.2\}$	0.2
neurones (n)	$\{20, 40, 60, 80, 100\}$	20
hidden (h)	$\{1, 2, 3, 4, 5\}$	3
composents (K)	$\{1, 2, 3, 4\}$	3

TABLE A.27 : Hyper-paramètres de MDN pour la garantie automobile (dommages).

Hyper-paramètres	Plage de valeurs	Valeur optimale
λ_w	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	10^{-3}
λ_σ	$\{0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$	10^{-4}
dropout (p)	$\{0, 0.1, 0.2\}$	0
neurones (n)	$\{20, 40, 60, 80, 100\}$	40
hidden (h)	$\{1, 2, 3, 4, 5\}$	3
composents (K)	$\{1, 2, 3, 4\}$	2

TABLE A.28 : Hyper-paramètres de MDN pour la garantie responsabilité civile générale.

A.2 La procédure Actuary-in-the-Box

Actuary-in-the-Box (Scarth (2020)) est une procédure générale d'estimation du risque lié aux provisions sur un an. Elle part du principe que nous disposons déjà d'une méthode algorithmique pour fixer les provisions, puis spécifie une procédure pour simuler l'évolution des sinistres de l'année suivante et réappliquer l'algorithme pour obtenir les provisions dans un an. La méthode est la suivante :

Étape 1 Obtenir la meilleure estimation (BE) des provisions d'ouverture. On suppose que cela est fait selon un algorithme bien défini, et qu'il n'inclut aucune marge de risque (RM).

Étape 2 Étendre les données d'entrée nécessaires pour l'algorithme utilisé à l'étape 1 en simulant une année supplémentaire de données.

Étape 3 Appliquer exactement le même algorithme qu'à l'étape 1 à l'ensemble de données étendu généré à l'étape 2 pour produire une distribution des provisions de sinistres de clôture.

Il est indéniable qu'il s'agit d'une approche assez générale, mais elle repose sur quelques hypothèses. En particulier, le processus de fixation des provisions est ou peut être approximé par un algorithme bien défini. Étant donné que les actuaires chargés de la gestion des provisions font généralement preuve de beaucoup de discernement lorsqu'ils fixent les provisions, il s'agit d'une hypothèse importante. La méthode suppose également que (à l'étape 2) les données de l'année suivante peuvent être simulées, et que l'algorithme peut ensuite être réappliqué à l'ensemble de données étendu (à l'étape 3). Cela peut ne pas être possible dans certains cas, par exemple si les provisions ont été modélisées de manière globale en utilisant des hypothèses de référence.

La situation la plus courante pour appliquer le principe de Actuary-in-the-Box est celle où on a bootstrapé un modèle pour obtenir la distribution finale des provisions. La méthode générale est la suivante :

Étape 1 Exécuter la procédure bootstrap .

Étape 2 Prolonger les données de sinistres d'une année en utilisant les résultats du bootstrap.

Étape 3 Réajustement du modèle déterministe sous-jacent aux données de sinistres étendues.

Étape 4 Calculer les sinistres ultimes pour les données de sinistres étendues en utilisant le modèle déterministe sous-jacent.

La procédure Actuary-in-the-Box est exactement la même pour le modèle de Mack ou le modèle GLM ODP. Nous commençons par un triangle de sinistres observés,

$$T = \{C_{i,j} : i = 1, ..., n; j = 1, ..., n - i + 1\},$$

représenté dans le schéma ci-dessous.

FIGURE A.1 : Triangle de sinistres observés.

L'étape 1 ci-dessus consiste à effectuer la procédure bootstrap pour obtenir des simulations de l'évolution future du triangle des sinistres,

$$T^* = \{C^*_{i,j} : i = 2, ..., n; j = n - i + 2, ..., n\},$$

Ceci est représenté dans la figure ci-dessous.

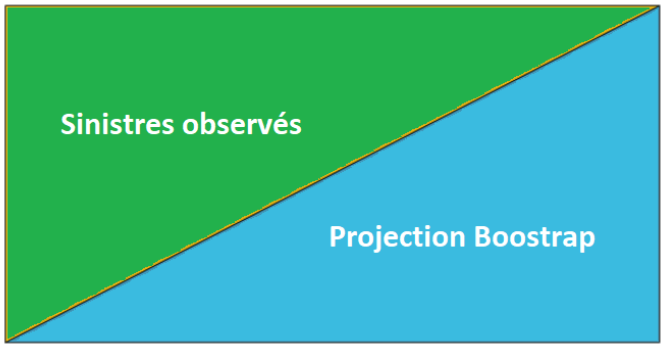

FIGURE A.2 : Triangle de sinistres observés et projection Boostrap.

L'étape 2 consiste à étendre les données de sinistres d'entrée en utilisant la sortie bootstrap. La sortie bootstrap correspondant à l'année suivante est la diagonale,

$$D^* = \left\{C^*_{i,n-i+2} : i = 2, \ldots, n\right\},$$

Le triangle original est étendu en ajoutant cette diagonale,

$$T' = \left\{C'_{ij} : i = 1, \ldots, n, j = 1, \ldots, n - i + 2\right\},$$

avec

$$C'_{ij} = \begin{cases} C_{ij} \text{ if } j \leq n - i + 1 \\ C^*_{ij} \text{ sinon} \end{cases}.$$

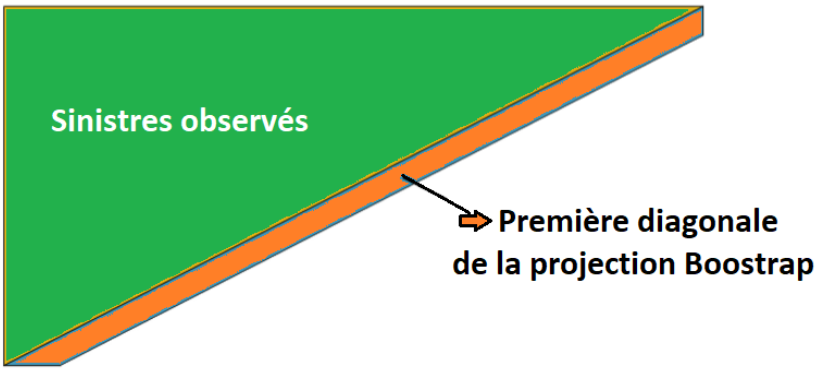

FIGURE A.3 : Triangle de sinistres observés avec la diagonale.

L'étape 3 consiste à réajuster le modèle déterministe sous-jacent aux données de sinistres étendues. Pour le modèle de Mack et le modèle de l'ODP, le modèle déterministe sous-jacent est le modèle Chain Ladder. Nous l'ajustons en calculant les facteurs de développement avec une légère modification des formules habituelles pour tenir compte de la diagonale supplémentaire.

$$\hat{f}'_j = \frac{\sum_{i=1}^{n-j+1} C'_{i,j+1}}{\sum_{i=1}^{n-j+1} C'_{ij}}.$$

Comme $C'_{i,n-i+2}$ est stochastique, les facteurs de développement ajustés $\hat{f}'_i$ sont également stochastiques.

L'étape 4 consiste à calculer l'estimation des sinistres ultimes pour les données de sinistres étendues en utilisant le modèle déterministe sous-jacent. Cela signifie simplement qu'il faut appliquer les facteurs de développement de la manière habituelle pour obtenir l'estimation des sinistres ultimes, en tenant compte de la diagonale supplémentaire.

$$\hat{C}'_{in} = C'_{i,n-i+2} \prod_{k=n-i+2}^{n-1} \hat{f}'_k.$$

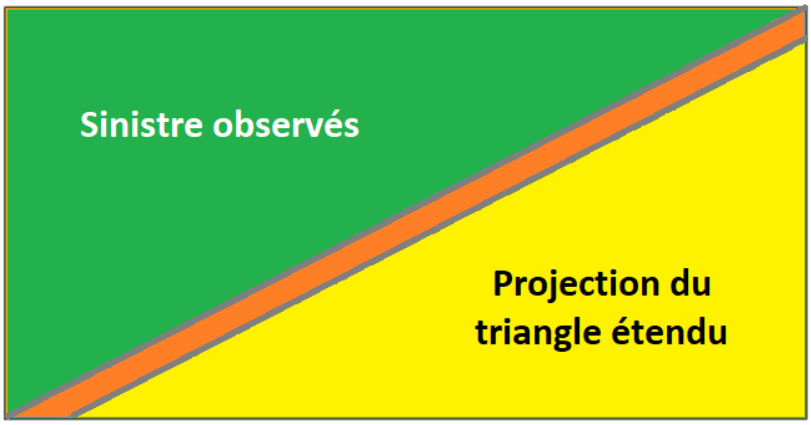

FIGURE A.4 : Triangle de sinistres observés étendu avec projection.

Si le triangle contient des données sur les sinistres payés, les provisions de clôture peuvent être calculées en utilisant la formule suivante : $R_i = \hat{C}'_{i,n} - C'_{i,n-i+2}$.

A.3 Exemples d'application des méthodes déterministes

A.3.1 Méthode de Chain Ladder

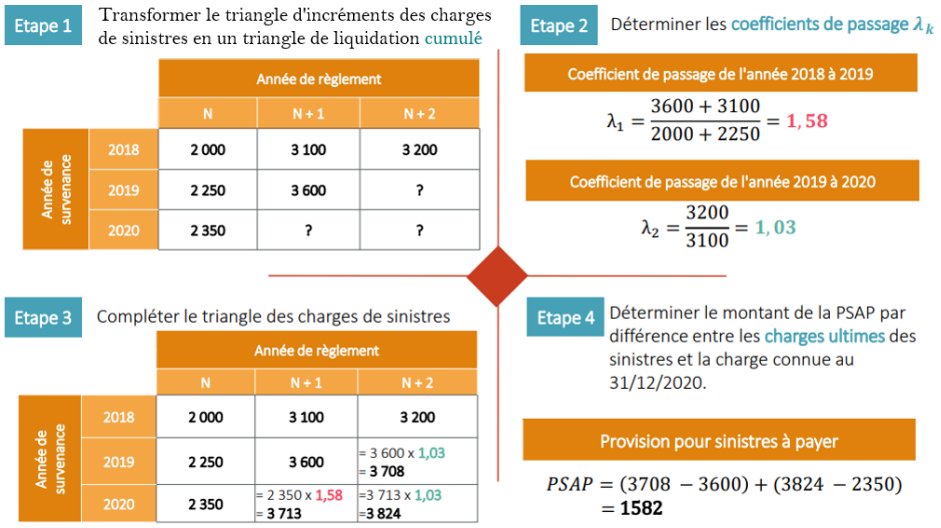

FIGURE A.5 : Exemple d'application de la méthode Chain Ladder.

A.3.2 Méthode de Bornhuetter-Ferguson

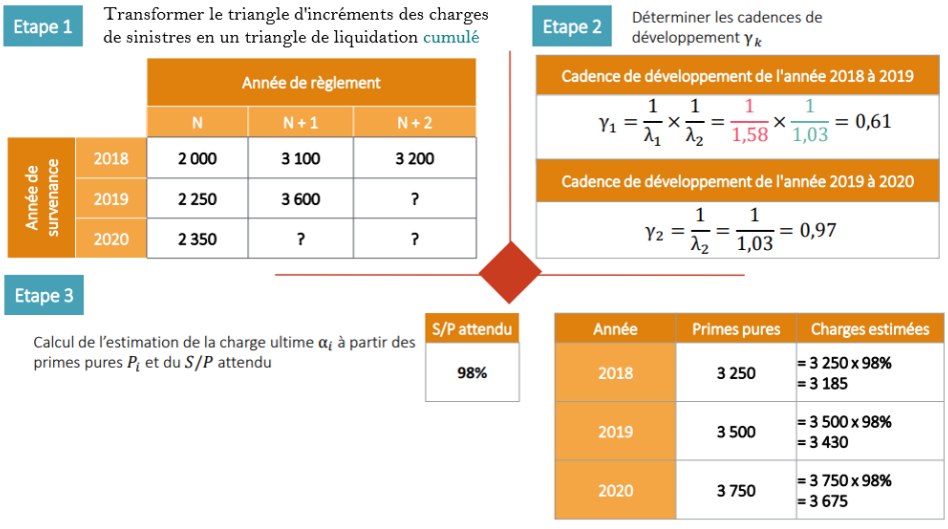

FIGURE A.6 : Exemple d'application de la méthode Bornhuetter-Ferguson.

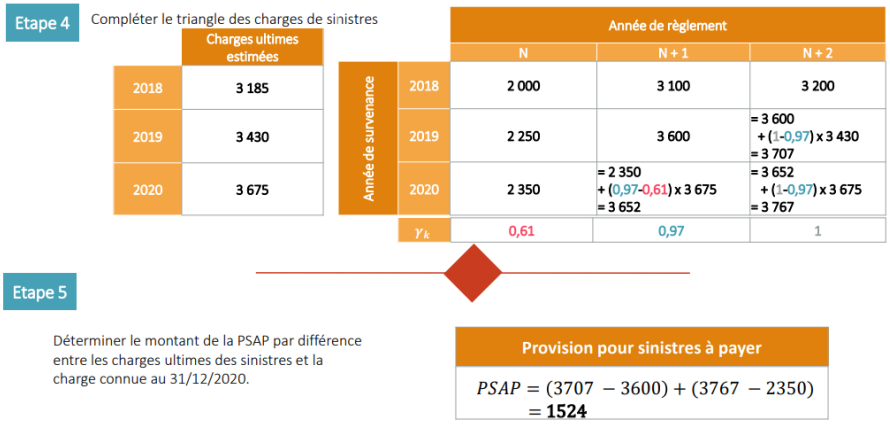

FIGURE A.7 : Exemple d'application de la méthode Bornhuetter-Ferguson.

A.4 Complément sur les Modèles Linéaires Généralisés (GLM)

A.4.1 Équations de vraisemblance

Soit f_Y la densité de probabilité de la variable Y. f_Y appartient à la famille exponentielle naturelle. La log vraisemblance de $Y \in \mathbb{R}^n$:

$$l(\beta) = \ln(f_{\theta,\gamma(\phi)}(Y)) = \sum_{i=1}^n \ln(f_{\theta_i,\gamma(\phi)}(y_i)) = \sum_{i=1}^n (\frac{1}{\gamma(\phi)}(y_i\theta_i - b(\theta_i)) + c(y_i,\phi)) = \sum_{i=1}^n l_i(\beta).$$

Les équations de vraisemblance sont :

$$\frac{\partial l(\beta)}{\partial \beta_i} = \sum_{i=1}^n (\frac{1}{Var(y_i|x_i)}(y_i - b'(x_i^T\beta))(g^{-1})'(x_i^T\beta)x_{i,j}) = 0\,;\ \forall j \in \{1,...,r\} \text{ avec } \theta_i = x_i^T\beta.$$

Sous forme matricielle, le gradient s'écrit :

$$\nabla l(\beta) = (\frac{\partial l(\beta)}{\partial \beta_1},...,\frac{\partial l(\beta)}{\partial \beta_r})^T = 0_r.$$

Lorsque le lien choisit est le lien canonique ou $g^{-1} = b'$, les équations de vraisemblance se simplifient de la facon suivante :

$$\sum_{i=1}^n x_i^j \frac{1}{\gamma(\phi)}(Y_i - b'(x_i\beta)) = \sum_{i=1}^n \frac{1}{\gamma(\phi)} x_i^j (Y_i - E[Y_i|x_i]) = 0\ ,\ \forall j \in \{1,...,r\}.$$

Les équations de vraisemblance n'ont pas de solution explicite en général, sauf dans le cas : $b'(\mu) = \mu$, ce qui correspond au modèle linéaire classique. On a donc recourt à des procédures d'optimisation itératives pour approcher la solution $\widehat{\beta}^{MV}$: *Algorithme IRLS/Newton-Raphson*, qui utilise les équations de transcendances.

A.4.2 Algorithme IRLS/Newton-Raphson

On applique le principe précédent à la dérivée $\beta \mapsto \nabla l(\beta)$ pour trouver un maximum local, l'algorithme de Newton-Raphson s'écrit de la facon suivante :

1. Choisir un point de départ $\beta^{(0)}$.

2. A l'itération $(k+1)$: calculer

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + A_k \nabla \mathcal{L}\left(\boldsymbol{\beta}^{(k)}\right),$$

avec

$$A_k = -\left[\mathcal{H}(\mathcal{L})\left(\beta^k\right)\right]^{-1}$$

la matrice Hessienne de $\mathcal{L}(\boldsymbol{\beta})$.

3. On s'arrête lorsque $\beta^{(k+1)} \approx \beta^{(k)}$ ou bien $\nabla \mathcal{L}\left(\beta^{k+1}\right) \approx \nabla \mathcal{L}\left(\beta^k\right)$.

A.4.3 Test de nullité des paramètres

En utilisant le résultat sur la loi asymptotique de $\widehat{\beta_n}$ il est direct d'écrire une statistique de test pour l'hypothèse $H_0 : \beta_k = 0$ vs $H_1 : \beta_k \neq 0$, En effet, la statistique :

$$T(Y_n) = \frac{\widehat{\beta_{n,k}}}{\sqrt{\widehat{s_k^2}}} \text{ avec } \widehat{s_k^2} = [-E[\mathcal{H}(\mathcal{L}(\beta))]]^{-1}_{k,k}$$

est de loi asymptotique normale centrée réduite sous l'hypothèse H_0. Par conséquent, on rejette H_0 si $|T| > q_{1-\frac{\alpha}{2}}$ avec $q_{1-\frac{\alpha}{2}}$ le quantile de niveau $1-\frac{\alpha}{2}$ d'une loi gaussienne centrée réduite.

A.4.4 Test de Wald

Test asymptotique de taille α pour $H_0 : \beta_k = 0$ vs $H_1 : \beta_k \neq 0$.

Sous $H_0 : S = T^2 \longrightarrow \chi^2_1$ lorsque $n \to +\infty$.

Sa zone de rejet est : $R_\alpha = \{S > q^{\chi^2_1}_{1-\alpha}\}$ avec $q^{\chi^2_1}_{1-\alpha}$ le quantile de niveau $1-\alpha$ de la loi χ^2_1.

A.4.5 Test d'un sous modèle

Soit le test asymptotique de taille α pour $H_0 : [m_0]$ de taille r_0 est adéquat vs $H_0 : [m_1]$ de taille r_1 est adéquat, avec $r_1 > r_0$ de statistique T :

$$T = 2(l_{[m_1]} - l_{[m_0]}).$$

Sous certaines hypothèses de régularité des modèles on a :

$$\text{Sous } H_0 : T \longrightarrow \chi^2_{r_1 - r_0} \text{ lorsque } n \to +\infty.$$

Sa zone de rejet : $R_\alpha = \{T > q^{\chi^2_{r_1-r_0}}_{1-\alpha}\}$ avec $q^{\chi^2_{r_1-r_0}}_{1-\alpha}$ le quantile de niveau $1-\alpha$ de la loi $\chi^2_{r_1-r_0}$.

A.5 Complément sur Decision Tree (Arbre de décision)

A.5.1 L'algorithme CART (Classification And Regression Trees)

L'algorithme CART utilisé dans l'arbre de décision, est une méthode non paramétrique permettant de construire des estimateurs d'une fonction de régression dans un cadre multidimensionnel. Les méthodes basées sur les arbres reposent sur une partition de l'espace des variables d'entrée, on infère ensuite un modèle simple sur chaque élément de la partition. Nous supposons que nous avons un échantillon de n $(\boldsymbol{X}_i, Y_i)_{1\leq i\leq n}$ avec $\boldsymbol{X}_i \in \mathbb{R}^d$ et $Y_i \in \mathbb{R}$. L'algorithme CART permet de définir, à partir de l'échantillon d'apprentissage, une partition automatique pilotée par les données de l'espace des variables d'entrée $\boldsymbol{X}_i$.

Supposons que l'espace des variables d'entrée $\boldsymbol{X}_i$ soit subdivisé en M régions, que nous désignons par $R_1, \dots R_M$. Nous introduisons la classe F de fonctions constantes par morceaux sur les éléments de la partition :

$$F = \left\{ f, f(\boldsymbol{x}) = \sum_{m=1}^{M} c_m \mathbf{1}_{\boldsymbol{x}\in R_m} \right\}.$$

L'estimateur des moindres carrés de la fonction de régression f sur la classe F minimise le critère

$$\sum_{m=1}^{M} \left(Y_i - f\left(\boldsymbol{X}_i\right)\right)^2,$$

parmi les fonctions $f \in F$. La solution est

$$\hat{f}(\boldsymbol{x}) = \sum_{m=1}^{M} \hat{c}_m \mathbf{1}_{\boldsymbol{x}\in R_m},$$

où $\hat{c}_m$ est la moyenne des observations Y_i telles que $\boldsymbol{X}_i \in R_m$. Afin de définir la partition, CART procède de la manière suivante :
Étant donné une variable de séparation $X^{(j)}$ et un point de séparation s, on considère les demi-espaces suivants $R_1(j,s) = \left\{\boldsymbol{X} = \left(X^{(1)}, \dots, X^{(d)}\right) / X^{(j)} \leq s\right\}$ et $R_2(j,s) = \left\{\boldsymbol{X} / X^{(j)} > s\right\}$. La variable de séparation $X^{(j)}$ et le point de séparation s sont choisis afin de résoudre le problème de minimisation :

$$\min_{j,s} \left[\sum_{i,\boldsymbol{X}_i\in R_1(j,s)} (Y_i - \hat{c}_1)^2 + \sum_{i,\boldsymbol{X}_i\in R_2(j,s)} (Y_i - \hat{c}_2)^2 \right].$$

Étant donné j et s, on partitionne les données dans les deux régions correspondantes, puis nous procédons à une nouvelle séparation sur chacune des deux sous-régions, et ainsi de suite, sur chaque sous-région obtenue. La taille de l'arbre est un paramètre à ajuster, qui est lié à la complexité de l'arbre. Un arbre avec un grand nombre de feuilles conduira à un sur-apprentissage et un arbre de petite taille conduira à un sous-apprentissage. Il est donc nécessaire de trouver la taille optimale de l'arbre avec une procédure adaptative, déterminée à partir des données. La stratégie consiste à construire un grand arbre et à élaguer l'arbre en minimisant un critère pénalisé. On note T un sous-arbre de T_0 si T peut être obtenu en réduisant le nombre de nœuds de T_0. Nous désignons par $|T|$ le nombre de nœuds terminaux de T et par $R_m, m = 1, \dots |T|$, la partition correspondant aux nœuds terminaux. Soit N_m le nombre d'observations pour lesquelles $\boldsymbol{X}_i \in R_m$. On a :

$$\hat{c}_m = \frac{1}{N_m} \sum_{i,\boldsymbol{X}_i\in R_m} Y_i,$$

et nous introduisons le critère à minimiser :

$$C_\lambda(T) = \sum_{m=1}^{|T|} \sum_{i, \boldsymbol{X}_i \in R_m} (Y_i - \hat{c}_m)^2 + \lambda |T|.$$

Pour tout λ, on peut prouver qu'il existe un unique arbre minimal T_λ minimisant le critère $C_\lambda(T)$. Pour trouver l'arbre T_λ, on élimine, à chaque étape, le nœud interne de l'arbre T qui réduit le moins le critère $\sum_m \sum_{i, \boldsymbol{X}_i \in R_m} (Y_i - \hat{c}_m)^2$. Cela donne une séquence de sous-arbres, qui contient l'arbre T_λ.

Le paramètre de régularisation λ doit également être calibré pour réaliser un bon compromis entre le biais et la variance de l'estimateur final.

A.6 Complément sur les Réseaux de neurones

A.6.1 Théorème d'approximation universelle

En 1991, Hornik (1991) a montré que toute fonction régulière et bornée $\mathbb{R}^d \to \mathbb{R}$ peut être approximée à une précision donnée par un réseau de neurones *feedforward* avec une couche cachée contenant un nombre fini de neurones, ayant la même fonction d'activation, et un neurone de sortie linéaire. Ce résultat a été prouvé précédemment par Cybenko (1989) dans le cas particulier de la fonction d'activation sigmoïde. Plus précisément, le théorème de Hornik peut être énoncé comme suit : *Soit ϕ une fonction d'activation bornée, continue et non décroissante. Soit K_d un ensemble compact dans $\mathbb{R}^d$ et $\mathcal{C}(K_d)$ l'ensemble des fonctions continues sur K_d. Soit $f \in \mathcal{C}(K_d)$. Alors pour tout $\varepsilon > 0$, il existe $N\mathbb{N}$, des nombres réels v_i, b_i et des vecteurs w_i de $\mathbb{R}^d$ tels que, si on définit :*

$$F(x) = \sum_{i=1}^{N} v_i \phi\left(\langle w_i, x \rangle + b_i\right),$$

alors nous avons : $\forall x \in K_d, |F(x) - f(x)| \leq \varepsilon$.

L'espace des réseaux de neurones *feedforward* est dense dans l'espace des fonctions mesurables (au sens de la mesure de Lebesgue) définies sur et à valeurs dans un espace de dimension fini. Ce théorème est intéressant d'un point de vue théorique. D'un point de vue pratique, il n'est pas vraiment utile car le nombre de neurones dans la couche cachée peut être très grand. La force de l'apprentissage profond réside dans la profondeur (nombre de couches cachées) des réseaux.

A.6.2 Rétropropagation pour la régression avec la perte quadratique

On considére le cas de la régression, dans ce qui suit, on expliquera comment calculer le gradient de la perte quadratique empirique par l'algorithme de la rétropropagation (*Backpropagation*). Pour simplifier, nous ne considérons pas ici le terme de pénalisation, qui peut facilement être ajouté. En supposant que la sortie du perceptron multicouche (PMC) est de taille K, et en utilisant les notations précedentes, la perte quadratique empirique est proportionnelle à : $\sum_{i=1}^{n} R_i(\theta)$ avec $R_i(\theta) = (Y_{i,k} - f_k(X_i, \theta))^2$.

Dans un problème de régression, la fonction d'activation de sortie ψ est généralement la fonction d'identité, pour être plus précis, on suppose que $\psi(a_1, \ldots, a_K) = (g_1(a_1), \ldots, g_K(a_K))$ où $g_1, \ldots, g_K$ sont des fonctions de $\mathbb{R}$ à $\mathbb{R}$. Calculons les dérivées partielles de R_i par rapport aux poids de la couche de sortie. En sachant que :

$$a^{(L+1)}(x) = b^{(L+1)} + W^{(L+1)} h^{(L)}(x),$$

on obtient :
$$\frac{\partial R_i}{\partial W_{k,m}^{(L+1)}} = -2\left(Y_{i,k} - f_k\left(X_i, \theta\right)\right) g_k'\left(a_k^{(L+1)}\left(X_i\right)\right) h_m^{(L)}\left(X_i\right).$$

En différentiant maintenant par rapport aux poids de la couche précédente :
$$\frac{\partial R_i}{\partial W_{m,l}^{(L)}} = -2\sum_{k=1}^{K}\left(Y_{i,k} - f_k\left(X_i, \theta\right)\right) g_k'\left(a_k^{(L+1)}\left(X_i\right)\right) \frac{\partial a_k^{(L+1)}\left(X_i\right)}{\partial W_{m,l}^{(L)}},$$

avec :
$$a_k^{(L+1)}(x) = \sum_j W_{k,j}^{(L+1)} h_j^{(L)}(x),$$
$$h_j^{(L)}(x) = \phi\left(b_j^{(L)} + \left\langle W_j^{(L)}, h^{(L-1)}(x)\right\rangle\right).$$

Cela conduit à :
$$\frac{\partial a_k^{(L+1)}(x)}{\partial W_{m,l}^{(L)}} = W_{k,m}^{(L+1)} \phi'\left(b_m^{(L)} + \left\langle W_m^{(L)}, h^{(L-1)}(x)\right\rangle\right) h_l^{(L-1)}(x).$$

Introduisons les notations :
$$\delta_{k,i} = -2\left(Y_{i,k} - f_k\left(X_i, \theta\right)\right) g_k'\left(a_k^{(L+1)}\left(X_i\right)\right),$$
$$s_{m,i} = \phi'\left(a_m^{(L)}\left(X_i\right)\right) \sum_{k=1}^{K} W_{k,m}^{(L+1)} \delta_{k,i},$$

alors on a :
$$\frac{\partial R_i}{\partial W_{k,m}^{(L+1)}} = \delta_{k,i} h_m^{(L)}\left(X_i\right), \tag{A.1}$$
$$\frac{\partial R_i}{\partial W_{m,l}^{(L)}} = s_{m,i} h_l^{(L-1)}\left(X_i\right). \tag{A.2}$$

Les valeurs du gradient sont utilisées pour mettre à jour les paramètres de l'algorithme de descente du gradient. À l'etape $r+1$, on a :
$$W_{k,m}^{(L+1,r+1)} = W_{k,m}^{(L+1,r)} - \varepsilon_r \sum_{i \in B} \frac{\partial R_i}{\partial W_{k,m}^{(L+1,r)}},$$
$$W_{m,l}^{(L,r+1)} = W_{m,l}^{(L,r)} - \varepsilon_r \sum_{i \in B} \frac{\partial R_i}{\partial W_{m,l}^{(L,r)}},$$

où B est le batch et $\varepsilon_r > 0$ est le taux d'apprentissage (learning rate) qui satisfait à $\varepsilon_r \to 0, \sum_r \varepsilon_r = \infty$, $\sum_r \varepsilon_r^2 < \infty$, par exemple $\varepsilon_r = \frac{1}{r}$. Nous utilisons les équations de rétropropagation pour calculer le gradient par un algorithme à deux pahses. Dans la phase précédente, nous fixons la valeur des poids actuels $\theta^{(r)} = \left(W^{(1,r)}, b^{(1,r)}, \ldots, W^{(L+1,r)}, b^{(L+1,r)}\right)$, et nous calculons les valeurs prédites $f\left(X_i, \theta^{(r)}\right)$ et toutes les valeurs intermédiaires $\left(a^{(k)}\left(X_i\right), h^{(k)}\left(X_i\right) = \phi\left(a^{(k)}\left(X_i\right)\right)\right)_{1 \le k \le L+1}$ qui sont mémorisées. En utilisant ces valeurs, nous calculons pendant la deuxième phase (le passage en arrière), les quantités $\delta_{k,i}$ et $s_{m,i}$ et les dérivées partielles données dans les équations (A.1) et (A.2).

Nous avons calculé les dérivées partielles de R_i uniquement par rapport aux poids de la couche de sortie et des précédentes, mais nous pouvons continuer à calculer les dérivées partielles de R_i par rapport aux poids des couches cachées précédentes. Dans l'algorithme de rétropropagation, chaque couche cachée donne et reçoit des informations des neurones avec lesquels elle est connectée. L'algorithme est donc adapté aux calculs parallèles.

A.7 Complément sur le modèle MDN

A.7.1 Apprentissage et séléction du modèle optimal

Le simulateur SynthETIC produit des sinistres individuels, qui sont regroupés dans un triangle de développement (40x40 trimestres). Il est fréquent, dans la modélisation des réseaux neuronaux, de normaliser les variables d'entrée afin de stabiliser l'apprentissage. Lors des premières étapes de l'expérimentation, la normalisation de la réponse $(X_{i,j})$ s'est avérée cruciale pour atteindre la convergence pendant la formation.

Pour chaque combinaison d'hyperparamètres testée $\Theta = \{\lambda_w, \lambda_\sigma, p, n, h, K\}$ avec $\boldsymbol{p}$ pour dropout, $\boldsymbol{n}$ pour nombre de neurones, $\boldsymbol{h}$ pour nombre de couches cachées et $\boldsymbol{K}$ pour le nombre de composants , un MDN avec ces hyperparamètres est entraîné sur l'ensemble d'apprentissage de chaque partition, puis projeté sur l'ensemble de test. La fonction de perte que nous avons choisie est la Log-Vraisemblance Négative (NLL), avec des pénalités d'activation de poids et de sigma appliquées pendant l'apprentissage :

$$\text{Loss}\left(\mathbf{X}, \hat{\mathbf{X}} \mid \mathbf{w}, \lambda_w, \lambda_\sigma\right) = -\frac{1}{|\mathbf{X}|} \sum_{i,j:X_{i,j}\in\mathbf{X}} \ln\left(f_{X_{i,j}}\left(X_{i,j} \mid \mathbf{w}\right)\right) + \lambda_w \mathbf{w}\cdot\mathbf{w} + \lambda_\sigma \sum_{i,j:X_{i,j}\in\mathbf{X}} \sum_{k=1}^{K} \sigma_{i,j,k}^2.$$

L'optimiseur Adam par l'expérimentation, avec un taux d'apprentissage de 0.001, a fourni l'apprentissage le plus stable par rapport aux autres optimiseurs comme RMSProp et Stochastic Gradient Descent. Pour minimiser davantage le sur-apprentissage, l'arrêt précoce a été utilisé pour arrêter l'apprentissage dès que la perte de validation était minimisée. La perte de validation a rarement diminué de manière constante, c'est pourquoi l'apprentissage n'a été arrêté que lorsqu'elle n'a pas atteint de nouveaux seuils dans les 1000 dernières époques. C'est ce que l'on appelle la mesure de patience dans l'interface Keras, une patience inférieure à 1000 arrêterait parfois prématurément l'apprentissage. L'apprentissage dure généralement plusieurs milliers d'époques, avec des taux d'abandon plus élevés et des réseaux plus grands nécessitant souvent de 10000 jusqu'à 15000 itérations. Une limite de 10000 époques a été fixée lors de l'exécution de l'algorithme d'optimisation des hyperparamètres, afin de renforcer l'efficacité.

A.6.1.1 L'erreur de test

Désignons par Θ les valeurs des hyper-paramètres du MDN en cours d'exécution. Soit $f_{\hat{X}_{i,j}}(x \mid \mathbf{w}, \Theta)$ la densité de $\hat{X}_{i,j}$ projetée par le MDN avec des hyper-paramètres θ et des poids $\mathbf{w}$. Soit T_1 et T_2 l'ensemble des cellules (i, j) de l'ensemble de test de la première et deuxième partition respectivement. Un MDN distinct est entraîné T fois sur chaque partition. Soit $\mathbf{w}_{p,t}$ les poids du t^{eme} modèle entraîné sur la p^{eme} partition. M. Taher Al-Mudafer (2021) souligne que l'erreur de test du MDN avec les hyperparamètres Θ est calculé à partir des équations suivants :

$$\text{Test Error}\left(\boldsymbol{\theta}, \text{ Partition 1}\right) = -\tfrac{1}{T|T1|}\sum_{t=1}^{T}\sum_{i,j:(i,j)\in T1} \ln\left(f_{\hat{X}_{i,j}}\left(X_{i,j} \mid \mathbf{w}_{1,t}, \boldsymbol{\theta}\right)\right),$$

$$\text{Test Error}\left(\boldsymbol{\theta}, \text{ Partition 2}\right) = -\tfrac{1}{T|T2|}\sum_{t=1}^{T}\sum_{i,j:(i,j)\in T2} \ln\left(f_{\hat{X}_{i,j}}\left(X_{i,j} \mid \mathbf{w}_{2,t}, \boldsymbol{\theta}\right)\right),$$

$$\text{TestError}(\boldsymbol{\theta}) = \tfrac{|T_1|*\text{TestError}(\boldsymbol{\theta},\text{Partition 1})+|T_2|*\text{TestError}(\boldsymbol{\theta},\text{Partition 2})}{|T_1|+|T_2|},$$

Par conséquent, le MDN est entraîné $2T$ fois pour chaque ensemble d'hyper-paramètres Θ, car chaque exécution a une initialisation de poids différente et donc un ajustement différent. La moyenne de l'erreur de ces exécutions réduit l'impact des initialisations de poids aléatoires sur la performance de l'ensemble d'hyper-paramètres Θ.

A.6.1.1 La méthode projection constraints

M. Taher Al-Mudafer (2021) a optimisé la méthode projection constraints pour contrôler directement les estimations des cellules du triangle inférieur, contrôler ainsi directement les projections effectuées par le MDN. cette méthode permet d'intégrer le jugement de l'expert, qui peut placer des limites (supérieures et inférieures) sur les estimations de tout ensemble souhaité de cellules (i, j) dans le triangle inférieur et pénaliser le MDN si ses estimations se situent en dehors de ces limites.

Soit C l'ensemble des cellules (i, j) du triangle inférieur, dont les projections sont soumises à des contraintes. Soit $C_{i,j}^{\text{inf}}$ et $C_{i,j}^{\text{sup}}$ les contraintes inférieure et supérieure des estimations centrales pour la cellule $(i, j) \in$ C. Soit $\hat{\mu}_{i,j} = E\left[\hat{X}_{i,j}\right]$. La fonction de perte pendant l'apprentissage suit :

$$\text{NLLLoss}(\mathbf{X}, \hat{\mathbf{X}} \mid \mathbf{w}) = -\frac{1}{|\mathbf{X}|} \sum_{i,j:X_{i,j} \in \text{ Apprentissage}} \ln\left(f_{\hat{X}_{i,j}}\left(X_{i,j} \mid \mathbf{w}\right)\right)$$
$$+ \text{ Regularisation } + \frac{\lambda_C}{|\mathbf{C}|} \sum_{i,j:(i,j)\in\mathbf{C}} \left[\max\left(0, \hat{\mu}_{i,j} - C_{i,j}^{\text{ sup}}\right)\right]^2 + \left[\max\left(0, C_{i,j}^{\text{inf}} - \hat{\mu}_{i,j}\right)\right]^2$$

,

Où Regularisation $= \lambda_w \mathbf{w} \cdot \mathbf{w} + \lambda_\sigma \sum_{i,j:X_{i,j}\in\mathbf{X}_{T \text{ rain}}} \sum_{k=1}^{K} \sigma_{i,j,k}^2$, et λ_C est un coefficient de pénalité de violation de contrainte.

Les contraintes appliquent une pénalité de distance carrée à la fonction de perte si l'estimation centrale des cellules contraintes dans le triangle inférieur viole les contraintes. Avec un coefficient de pénalité suffisamment élevé, la projection du MDN satisfait les contraintes spécifiées, fournissant des projections plus raisonnables. Les cellules de C sont divisées aléatoirement en deux entre les ensembles d'apprentissage et de validation, car la perte de validation devrait indiquer dans quelle mesure les contraintes de projection ont été respectées pour que l'arrêt précoce soit utilisé efficacement.

A.8 Charges cumulées des environnements simulés

La figure suivante représente le flux de la charge cumulée des trimestres d'accident en fonction des trimestres de développement pour le triangle supérieur pour chaque environnement :

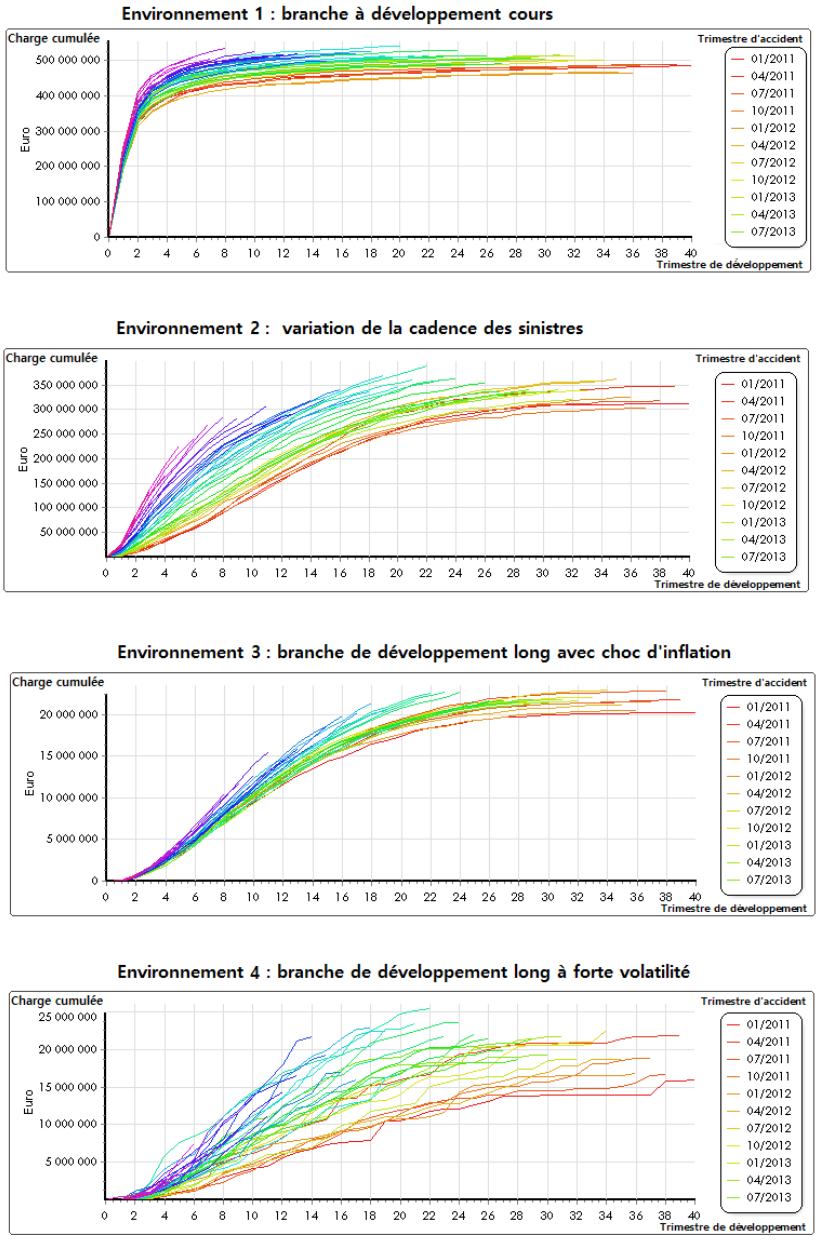

FIGURE A.8 : Représentation des charges cumulées en focntion des trimestres de développement pour chaque environnement.

A.9 Complèment sur l'analyse et comparaison de résultats

Les figures suivantes représentent les prédictions de la charge incrémentale des trimestres d'accident en fonction des trimestres de développement afin de zoomer sur le comportement de chaque modèle. Les graphes suivant représente les prédictions de chaque modèles. On se limite de présenter seulement le ***8ème***, ***16ème***, ***32ème*** et le ***40ème*** trimetre d'accident (AQ). Ce choix permet de bien analyser les valeurs prédites par rapport au valeurs simulées à court terme, au moyen terme et à long terme.

Le réctangle gris à gauche représente les valeurs du triangle supérieur (*le passé*), tandis que le triangle blanc à droite représente les valeurs du triangle inférieur (*le futur ou les valeurs prédites*).

A.9.1 Environnement 1 : branche à développement court

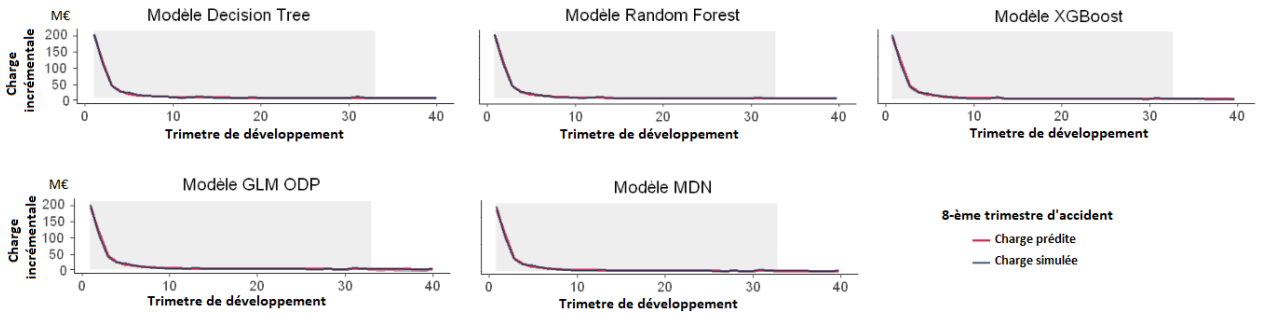

FIGURE A.9 : Graphe de prédictions des modèles avec *AQ=8* pour l'environnement 1.

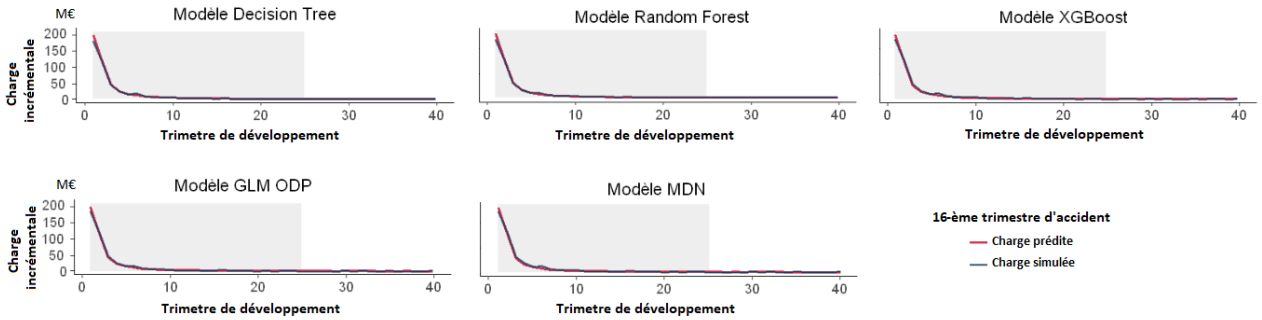

FIGURE A.10 : Graphe de prédictions des modèles avec *AQ=16* pour l'environnement 1.

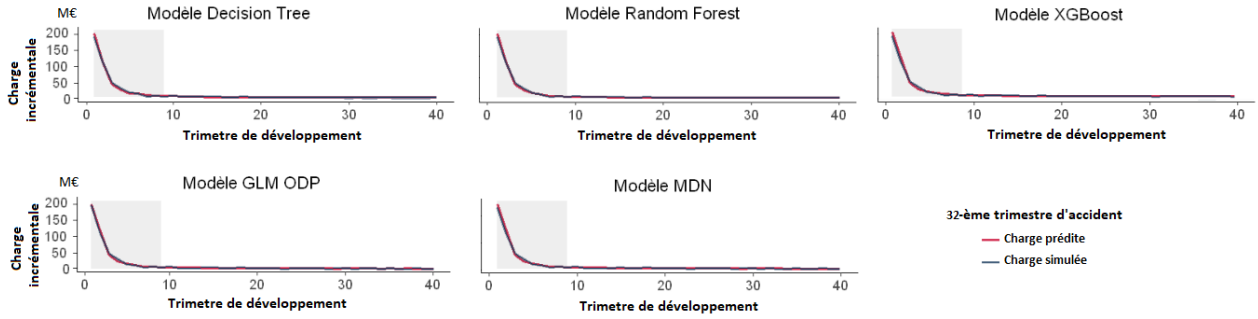

FIGURE A.11 : Graphe de prédictions des modèles avec AQ=*32* pour l'environnement 1.

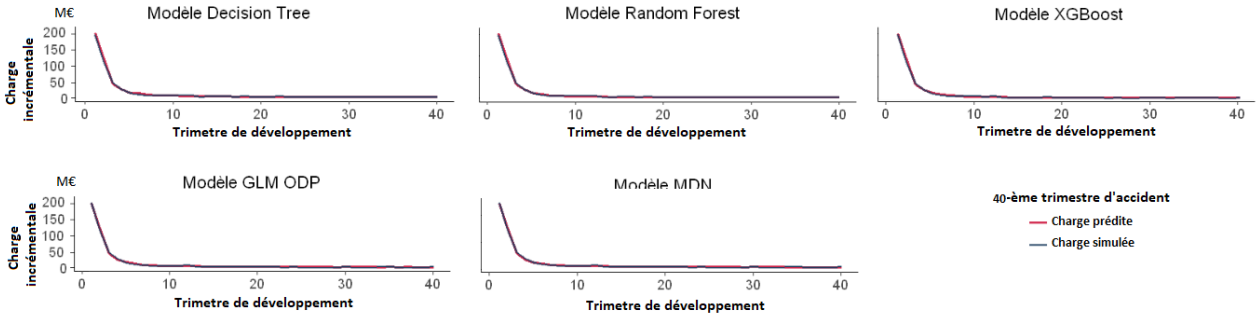

FIGURE A.12 : Graphe de prédictions des modèles avec AQ=*40* pour l'environnement 1.

A.9.2 Environnement 2 : variation de la cadence des sinistres

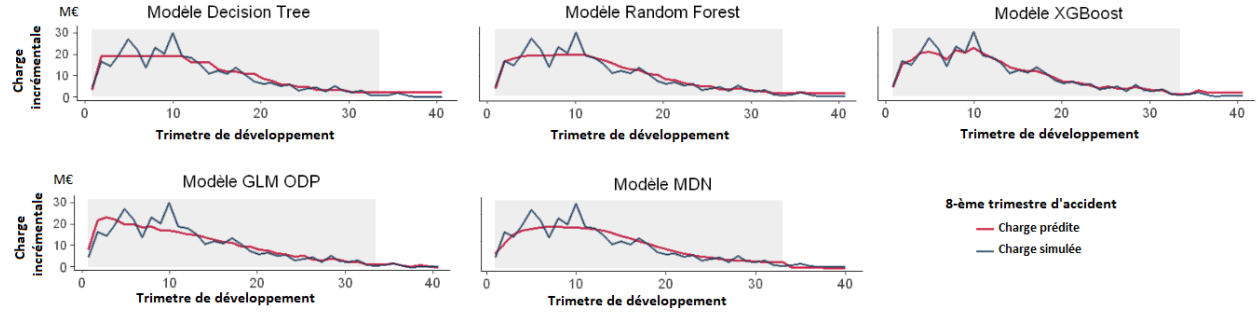

FIGURE A.13 : Graphe de prédictions des modèles avec AQ=*8* pour l'environnement 2.

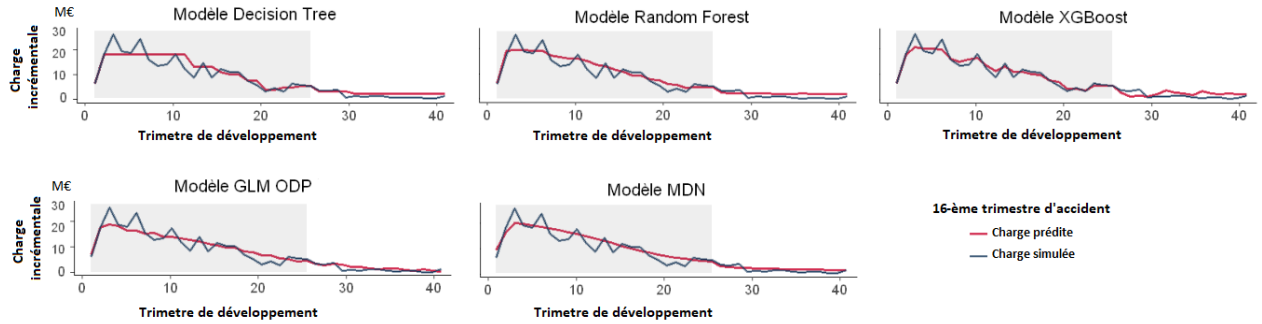

FIGURE A.14 : Graphe de prédictions des modèles avec *AQ=16* pour l'environnement 2.

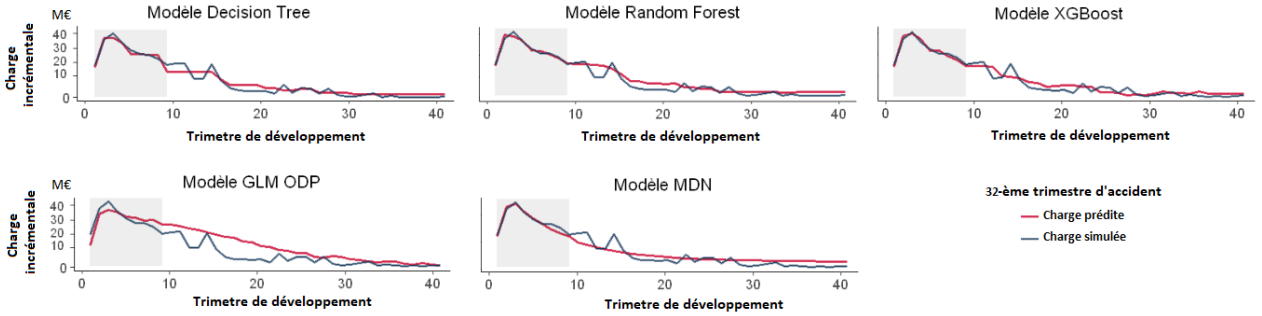

FIGURE A.15 : Graphe de prédictions des modèles avec *AQ=32* pour l'environnement 2.

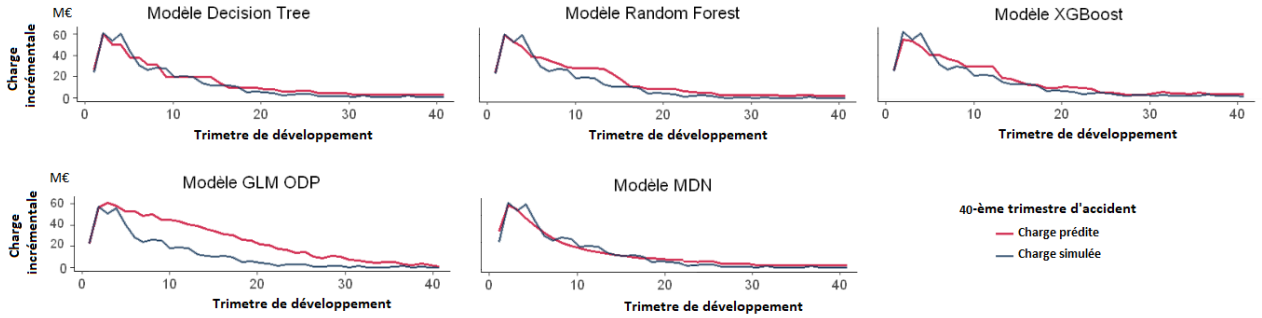

FIGURE A.16 : Graphe de prédictions des modèles avec *AQ=40* pour l'environnement 2.

A.9.3 Environnement 3 : branche à développement long avec un choc d'inflation

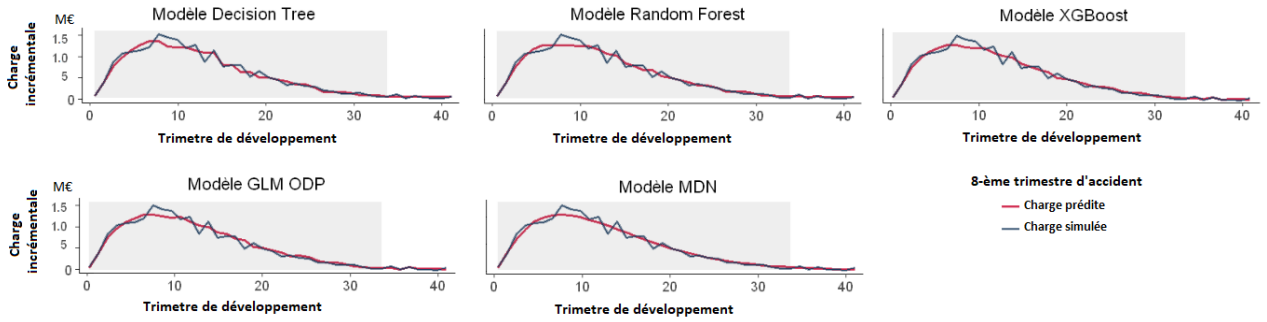

FIGURE A.17 : Graphe de prédictions des modèles avec AQ=*8* pour l'environnement 3.

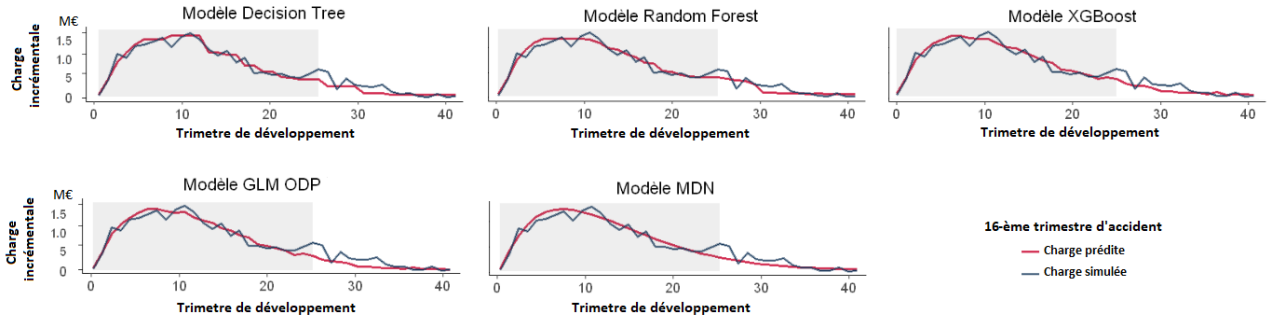

FIGURE A.18 : Graphe de prédictions des modèles avec AQ=*16* pour l'environnement 3.

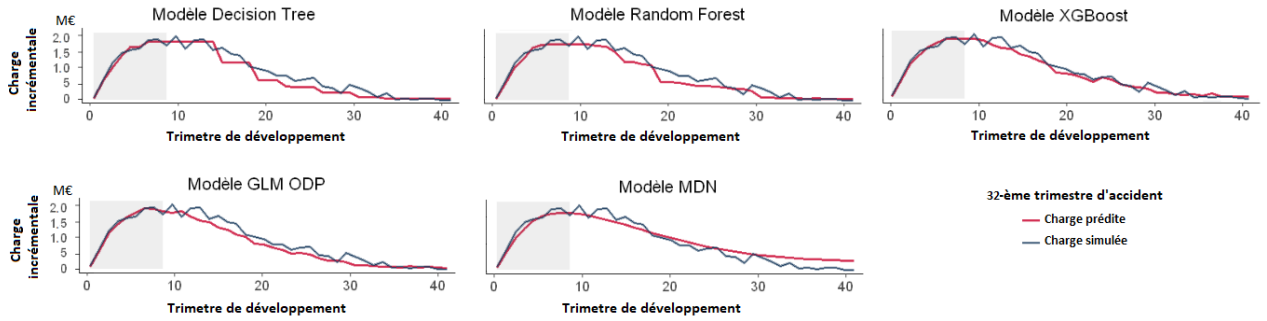

FIGURE A.19 : Graphe de prédictions des modèles avec AQ=*32* pour l'environnement 3.

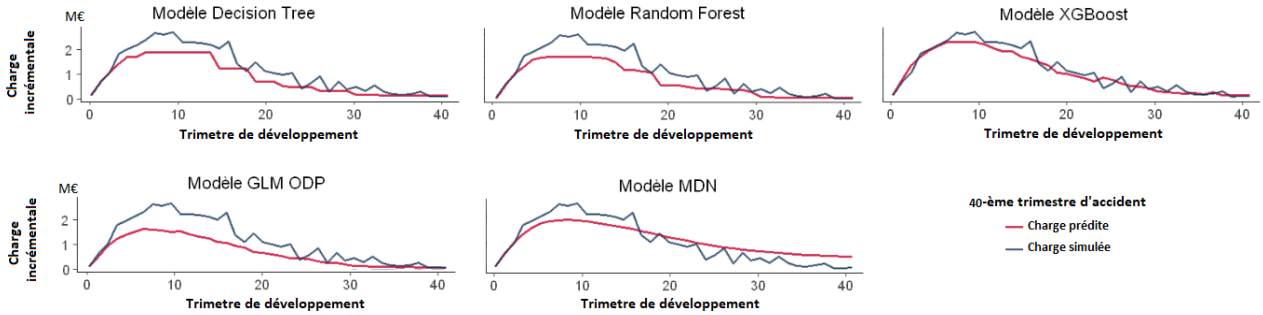

FIGURE A.20 : Graphe de prédictions des modèles avec AQ=*40* pour l'environnement 3.

A.9.4 Environnement 4 : branche à développement long à forte volatilité

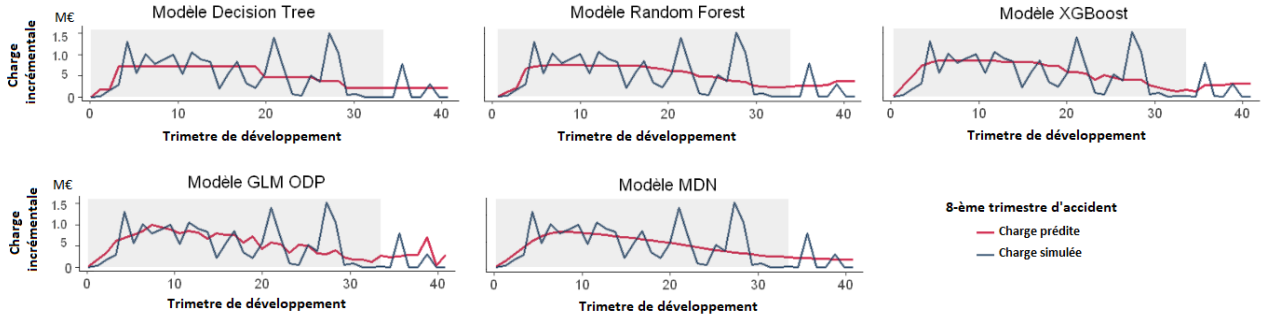

FIGURE A.21 : Graphe de prédictions des modèles avec AQ=*8* pour l'environnement 4.

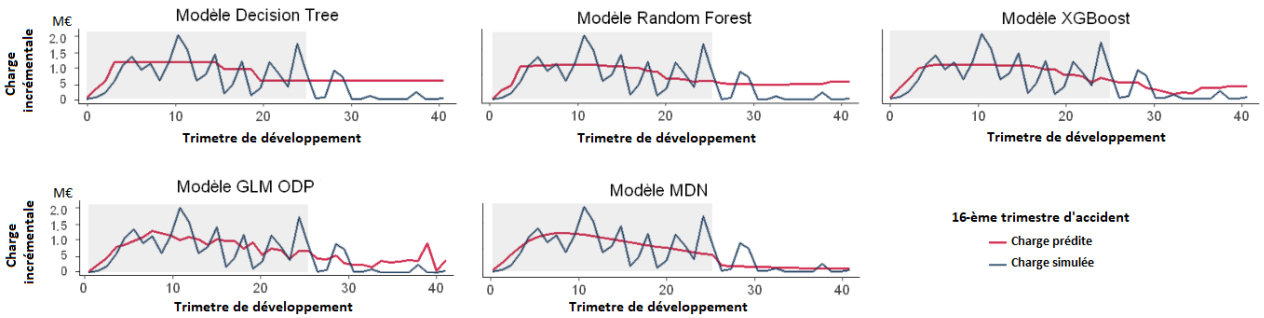

FIGURE A.22 : Graphe de prédictions des modèles avec *AQ=16* pour l'environnement 4.

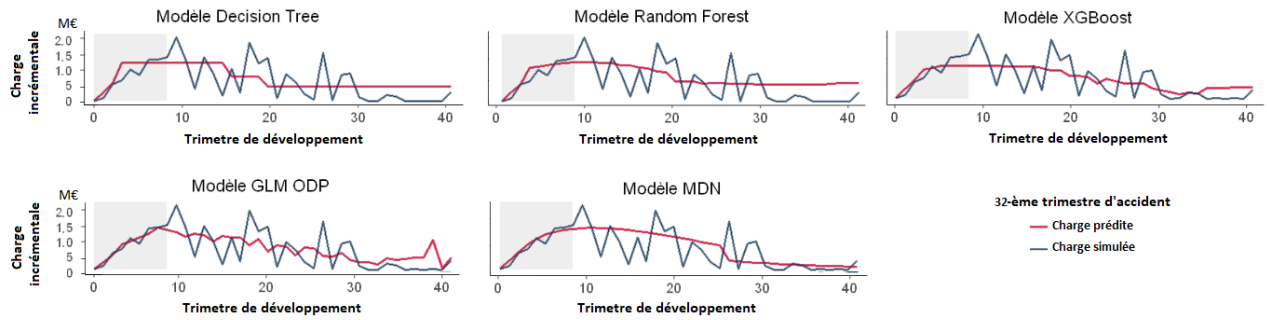

FIGURE A.23 : Graphe de prédictions des modèles avec *AQ=32* pour l'environnement 4.

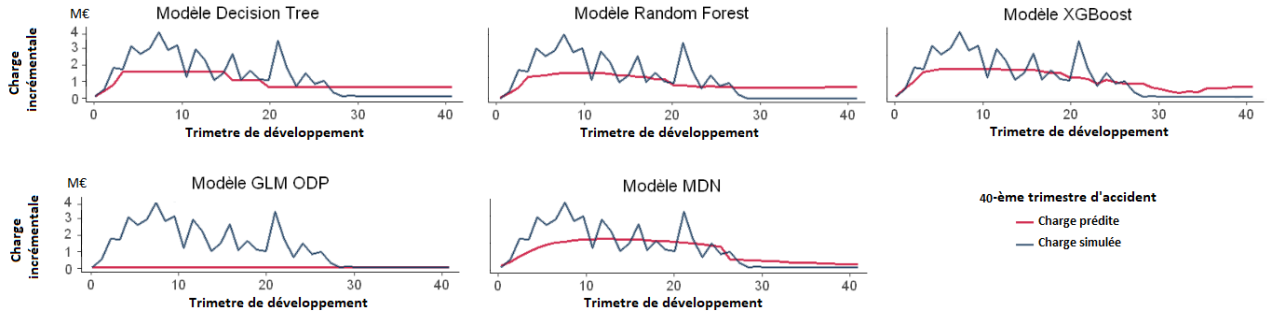

FIGURE A.24 : Graphe de prédictions des modèles avec *AQ=40* pour l'environnement 4.