Master Actuariat de Dauphine

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

le _____16/10/2017_____

Par : Paul OTTOU

Titre: Méthodes d'apprentissage automatique appliquées au provisionnement ligne à ligne en assurance non-vie

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

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

Membre présent du jury de l'Institut des Actuaires :

Signature :

Entreprise :

Nom : PricewaterhouseCoopers (PwC)

Signature :

Directeur de mémoire en entreprise :

Nom : Alexandre VEBER

Signature :

Membres présents du jury du Master Actuariat de Dauphine :

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

Signature du responsable entreprise :

Secrétariat :

Bibliothèque :

Signature du candidat :

Université Paris-Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 PARIS Cedex 16

Résumé

Aujourd'hui le provisionnement, très souvent volatile, rend difficile la communication des entreprises concernant leurs résultats (qu'ils soient financiers ou liés à des indicateurs de solvabilité). La volatilité peut s'expliquer notamment par des arbitrages à effectuer dans le choix des hypothèses de provisionnement. Dans le même temps, les réglementations prudentielles telle que Solvabilité 2 souhaitent une gestion des engagements toujours plus prudente et une communication toujours plus transparente.

À l'heure actuelle du « tout numérique », des quantités pharaoniques de données [1] sont recueillies, et ce, quel que soit le domaine. Le monde de l'assurance n'échappe pas à cette véritable révolution numérique. Dans ce contexte récent de collecte massive d'information, les algorithmes de machine learning s'avèrent être des outils très utiles et adaptés à l'analyse de toutes ces données. Et les assureurs, détenteurs de grandes quantités de données en tout genre, voient dans cette révolution une opportunité formidable d'exploiter leurs ressources en données afin d'améliorer la gestion de leurs risques et établir une meilleure stratégie de provisionnement ni trop optimiste (entraînant une sous-évaluation du risque) ni trop pessimiste (entraînant une immobilisation non nécessaire de capitaux).

L'objectif de ce mémoire est de réfléchir à des approches utilisant le plus d'information possible sur les sinistres afin d'obtenir des estimations de provision moins volatiles et plus précises que les estimations faites avec des méthodes de provisionnement traditionnelles comme Chain-Ladder. Mais outre la fiabilité du calcul, l'intérêt des nouvelles méthodes est de pouvoir individualiser le provisionnement. Ce serait un pas en avant comparé aux méthodes traditionnelles qui fonctionnent en agrégeant les sinistres. Ainsi, ce projet se base sur des données issues de l'activité Garantie contre les accidents de la vie (appelée GAV ci-après). Dans un premier temps, les provisions sont développées à partir de méthodes agrégées classiques. Un rappel de ces méthodes est proposé au lecteur. Les premiers résultats de provision serviront alors de point de comparaison des performances des autres méthodes.

Dans une seconde partie, deux méthodes sont présentées et utilisées dans le cadre du provisionnement ligne à ligne. La première méthode est l'algorithme de gradient boosting XGBoost. La seconde approche est également issue du machine learning, il s'agit du Long Short Term Memory (abrégé LSTM).

Dans une dernière partie, la conformité de ces nouvelles méthodes à la norme réglementaire IFRS est évaluée. Il s'agit de discuter de l'apport possible de ces algorithmes au cadre défini par IFRS 17.

Mots clés : *Data Mining, Machine learning, IFRS, Assurance non-vie, Provisionnement, Apprentissage automatique, Réseau de neurones, Chain-Ladder, LSTM, XGBoost*

1. D'un point de vue informatique, il s'agit une description élémentaire, souvent codée, d'une réalité (chose, fait, évènement...)

Abstract

Today, reserving, which is very often volatile, makes it difficult for companies to communicate about their results (either their financial results or the results linked to solvency indicators). Volatility can be explained in particular by arbitrations in the choice of reserving assumptions. At the same time, prudential regulations such as Solvency II require an ever more cautious management of commitments and an ever more transparent communication.

Nowadays, during the « digital era », pharaonic quantities of data [2] are recorded, whatever the field. The world of insurance does not escape this true digital revolution. In this recent context of massive information gathering, machine learning algorithms are supposed to be very useful tools adapted to the analysis of all these data. And insurance companies, with large amounts of data of all kinds, consider this revolution as a tremendous opportunity to exploit their online resources in order to improve risk management and establish a better strategy of reserving neither too optimistic (leading to undervaluation of the risk) nor too pessimistic (leading to unnecessary trappped capital).

The objective of this thesis is to consider approaches using as much information as possible about claims, in order to obtain less volatile and more accurate reserve estimates than estimates made with traditional reserving methods such as Chain-Ladder. Besides the reliability of the calculation, the interest of the new methods is to be able to individualize the reserving. This would be a step forward compared to traditional methods that work by aggregating claims. Thus, this study is based on data from the Life Accident Assurance activity (hereafter referred to as GAV). Initially, the provisions are developed using classical aggregate methods. A reminder of the methods is proposed to the reader. This firsts results will then serve as a point of comparison of the performances of the other methods.

In a second part, two methods are presented and used in line-by-line reserving. The first method is the XGBoost boosting gradient algorithm. The second approach is also based on machine learning, the Long Short-Term Memory (abbreviated LSTM)

In a final section, the conformity of these new methods with the IFRS regulatory standard is evaluated. The possible contribution of these algorithms to the framework defined by IFRS 17 will be discussed.

Keywords : *Data Mining, Machine learning, IFRS, Non-life Insurance, Reserving, Automatic Learning, Neural network, Chain-Ladder, LSTM, XGBoost*

2. From a computer perspective, this is an elementary description, often coded, of a reality (chosen, fact, event ...)

Note de synthèse

Mots clés : *Data Mining, Machine learning, IFRS, Assurance non-vie, Provisionnement, Apprentissage automatique, Réseau de neurones, Chain-Ladder, LSTM, XGBoost*

Introduction

Aujourd'hui le provisionnement, très souvent volatile, rend difficile la communication des entreprises concernant leurs résultats (qu'ils soient financiers ou liés à des indicateurs de solvabilité). La volatilité peut s'expliquer notamment par des arbitrages à effectuer dans le choix des hypothèses de provisionnement. Dans le même temps, les réglementations prudentielles telles que Solvabilité 2 souhaitent une gestion des engagements toujours plus prudente et une communication toujours plus transparente.

À l'heure actuelle du « tout numérique », des quantités pharaoniques de données[3] sont recueillies, et ce, quel que soit le domaine. Le monde de l'assurance n'échappe pas à cette véritable révolution numérique. Dans ce contexte récent de collecte massive d'information, les algorithmes de machine learning s'avèrent être des outils très utiles et adaptés à l'analyse de toutes ces données. Et les assureurs, détenteurs de grandes quantités de données en tout genre, voient dans cette révolution une opportunité formidable d'exploiter leurs ressources en données afin d'améliorer la gestion de leurs risques et donc de mieux répondre aux exigences réglementaires. De même, l'utilisation de ces informations devrait permettre d'établir une meilleure stratégie de provisionnement ni trop optimiste (entraînant une sous-évaluation du risque) ni trop pessimiste (entraînant une immobilisation non nécessaire de capitaux). L'utilisation croissante du machine learning semble donc marquer le début d'une nouvelle ère dans le secteur de l'assurance.

Enfin, outre la fiabilité du calcul, l'intérêt des nouvelles méthodes est de pouvoir individualiser le provisionnement. Le but de ce mémoire est également de s'interroger sur la manière dont il est possible d'avoir pour chaque sinistre, une vision à l'ultime de ce qu'il coûtera à l'entreprise. Ce serait un pas en avant comparé aux méthodes traditionnelles qui fonctionnent en agrégeant les sinistres. Et en agrégeant, ces méthodes traditionnelles perdent de l'information. Elles ne prennent pas en compte l'hétérogénéité du développement des sinistres et n'utilisent pas non plus tous les éléments sans doute corrélés à l'évolution de la sinistralité.

Ainsi, ce mémoire propose un rappel sur les méthodes historiques dans une première partie, ces premiers résultats servant de point de comparaison pour tester la performance des autres méthodes. Dans une deuxième partie, deux approches sont présentées et utilisées dans le cadre du provisionnement ligne à ligne. La première méthode est l'algorithme de gradient boosting XGBoost. La seconde approche est également issue du machine learning, il s'agit du Long Short-Term Memory (abrégé LSTM). Enfin, dans une troisième partie, ce mémoire s'intéresse au possible apport de ces nouvelles approches au cadre défini par IFRS 17.

Un grand soin a d'abord été accordé à l'analyse et au traitement de la base de données, l'objectif étant d'avoir dans la base finale une version de chaque sinistre par trimestre durant les cinq premières années de son développement (soit 20 trimestres).

3. D'un point de vue informatique, il s'agit une description élémentaire, souvent codée, d'une réalité (chose, fait, événement...)

À partir de cette base globale, une base de test a été construite avec les sinistres ayant une année de survenance entre 2002 et 2012 (en effet, la charge ultime de ces sinistres est, par hypothèse, connue à juin 2017, i.e. à aujourd'hui). On comparera alors les estimations de provision réalisées sur cette base avec les trois méthodes présentées par la suite.

Enfin, le point de départ de la projection est la vue à fin 2012 de la charge totale des sinistres de l'échantillon test.

La méthode de Chain-Ladder

La technique de Chain-Ladder permet de projeter le montant de la charge de sinistres en utilisant des facteurs de développement. Les sinistres sont pour cela agrégés par année de survenance.

Dans un premier temps, les montants de provision estimés pour les sinistres ayant une année de survenance entre 2002 et 2012 sont comparés aux montants des provisions réelles observées, i.e. le montant de provision que la compagnie d'assurance devait possédait pour faire face à ses engagements au cours des onze exercices. Ce montant est observé aujourd'hui, à savoir juin 2017.

Année de survenance	Provision observée	Provision estimée avec Chain-Ladder
2002	-11 970 €	-8 487 €
2003	0 €	-69 830 €
2004	-488 232 €	-134 225 €
2005	-1 652 €	-79 148 €
2006	-454 292,00 €	-336 112 €
2007	-1 138 121 €	-564 787 €
2008	-461 045 €	-300 691 €
2009	-692 705 €	-392 928 €
2010	-1 782 887 €	-270 554 €
2011	-969 772 €	330 231 €
2012	1 154 722 €	1 077 049 €
Total	**-4 845 953 €**	**-749 482 €**

montants en euros

FIGURE 1 – Analyse détaillée des provisions estimées pas Chain-Ladder avec les valeurs réelles

On constate que dans la réalité la charge de ses sinistres a baissé, entraînant des montants de provision négatifs.

Sur le montant de provision total, Chain-Ladder commet une erreur de plus de 85%. La méthode classique a en effet beaucoup de difficultés à estimer les montants de provision les plus importants comme c'est le cas en 2010 et 2011. Ainsi, même si sa simplicité de mise en place et son auditabilité en font un outils très pratique, la méthode de Chain-Ladder semble donc montrer quelques difficultés à estimer les provisions dès lors que les sinistres n'ont pas tous un développement homogène.

Estimation des provisions à l'aide de la méthode XGBoost

L'algorithme XGBoost est un algorithme ensembliste qui agrège des arbres. À chaque itération, le nouvel arbre apprend de l'erreur commise par l'arbre précédent. Ainsi, même si chaque arbre a un pouvoir prédictif faible, la règle de décision construite en sommant le résultat de chaque arbre est elle très fiable. Dans le cadre de notre problématique, ce genre d'algorithme a l'avantage de pouvoir utiliser toute l'information dont on dispose sur un sinistre.

La charge de chaque sinistre a été projetée à l'ultime en utilisant cet algorithme. Il a ensuite été possible d'en déduire la provision estimée par l'approche en comparant le montant de la charge projetée à l'ultime et le montant de la charge au moment de la projection. Les résultats de ces provisions estimées sont comparés à ceux de Chain-Ladder (présentés précédemment) et aux provisions réelles observées.

Année de survenance	Provision observée	Provision estimée avec Chain-Ladder	Provision estimée avec XGBoost
2002	-11 970 €	-8 487 €	339 870 €
2003	0 €	-69 830 €	-164 520 €
2004	-488 232 €	-134 225 €	-314 675 €
2005	-1 652 €	-79 148 €	24 773 €
2006	-454 292,00 €	-336 112 €	-596 011 €
2007	-1 138 121 €	-564 787 €	-1 071 289 €
2008	-461 045 €	-300 691 €	-39 057 €
2009	-692 705 €	-392 928 €	-445 495 €
2010	-1 782 887 €	-270 554 €	-1 073 863 €
2011	-969 772 €	330 231 €	-1 655 465 €
2012	1 154 722 €	1 077 049 €	-15 950 €
Total	**-4 845 953 €**	**-749 482 €**	**-5 011 681 €**

montants en euros

FIGURE 2 – Comparaison entre les charges réelles et les charges estimées par Chain-Ladder et XGBoost

La projection à l'aide de XGBoost permet de diminuer de manière drastique l'erreur globale qui n'est plus que de 3% alors qu'elle était de 85% pour Chain-Ladder.

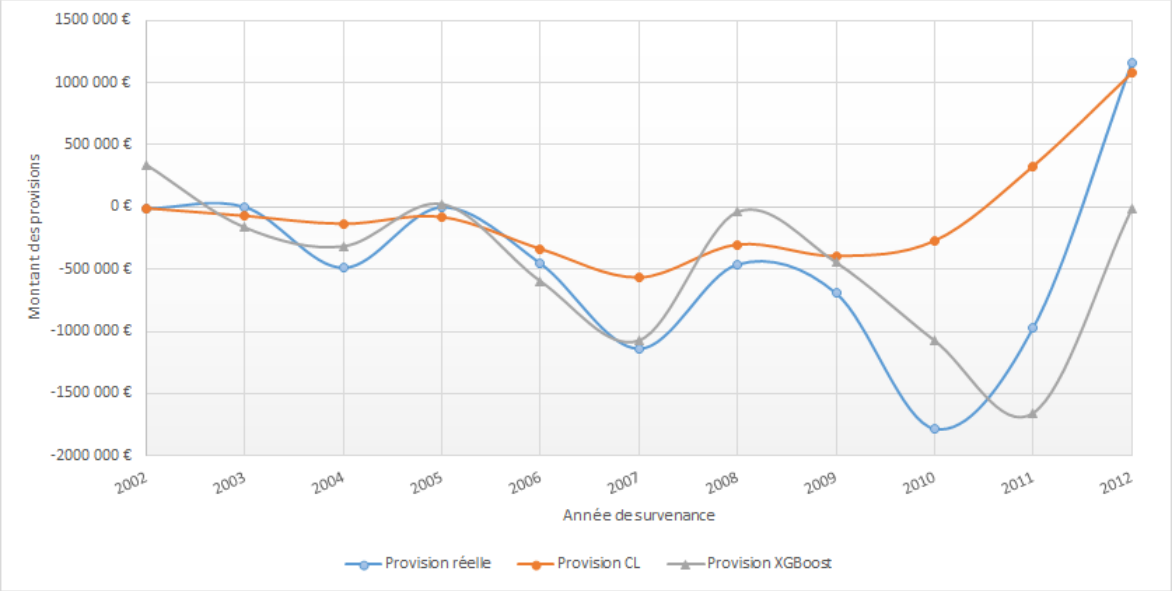


FIGURE 3 – Comparaison de l'estimation des provisions avec Chain-Ladder et XGBoost par année de survenance

Graphiquement on constate que jusqu'en 2009 les courbes sont très proches. Cependant sur les années de survenance les plus récentes, excepté 2012, l'utilisation de l'algorithme XGBoost permet d'avoir des estimations de provision bien plus proches du montant des provisions observées que celles de Chain-Ladder.

Enfin, l'algorithme XGBoost donne la possibilité de visualiser les variables qu'il a le plus utilisé pour construire sa règle de décision finale. Dans les cas de cette étude, parmi les trois variables les plus significatives se trouvent les variables correspondant à la charge : *la provision* et le *montant des paiements cumulés*.

Estimation des provisions à l'aide de la méthode Long Short-Term Memory

L'algorithme Long Short-Term Memory (abbrégé LSTM) fait partie de la famille des réseaux de neurones. Sa spécificité est de posséder une mémoire. En effet, grâce à un système de « cloisons » il est capable de lire, écrire ou effacer les informations qui lui ont été fournies au cours de ses différentes itérations. L'utilisation de cette approche est pertinente dans notre cas puisque l'on considère que le montant de la charge ultime d'un sinistre est lié aux différents évènements qui ont eu lieu au cours de sa vie.

Ainsi, comme pour les deux approches précédentes, l'écart entre les provisions réellement observées et les provisions estimées sur les exercices de 2002 à 2012 a été calculé.

Année de survenance	Provision réelle	Provision CL	Provision XGBoost	Provision LSTM
2002	-11 970 €	-8 487 €	339 870 €	2 507 240 €
2003	0 €	-69 830 €	-164 520 €	-26 563 €
2004	-488 232 €	-134 225 €	-314 675 €	-439 624 €
2005	-1 652 €	-79 148 €	24 773 €	538 649 €
2006	-454 292,00 €	-336 112 €	-596 011 €	-248 235 €
2007	-1 138 121 €	-564 787 €	-1 071 289 €	-366 879 €
2008	-461 045 €	-300 691 €	-39 057 €	-79 211 €
2009	-692 705 €	-392 928 €	-445 495 €	-641 277 €
2010	-1 782 887 €	-270 554 €	-1 073 863 €	-1 829 730 €
2011	-969 772 €	330 231 €	-1 655 465 €	-2 134 382 €
2012	1 154 722 €	1 077 049 €	-15 950 €	294 528 €
Total	**-4 845 953 €**	**-749 482 €**	**-5 011 681 €**	**-2 425 484 €**

montants en euros

FIGURE 4 – Analyse de l'erreur d'estimation des provisions avec Chain-Ladder, XGBoost et LSTM

Sur l'estimation de la provision globale, c'est l'approche à base d'arbres XGBoost qui est la plus précise. En effet, c'est elle qui est la plus proche de la provision globale réelle observée à posteriori. L'erreur commise par XGBoost est d'un peu plus de 3% (soit 300 000 euros environ). Ensuite, c'est l'algorithme LSTM qui est le plus proche de la valeur observée avec un écart de plus de 2,4 millions d'euros tout de même. C'est donc l'approche classique par Chain-Ladder qui est la plus éloignée du résultat global (plus de 4 millions d'euros de différence).

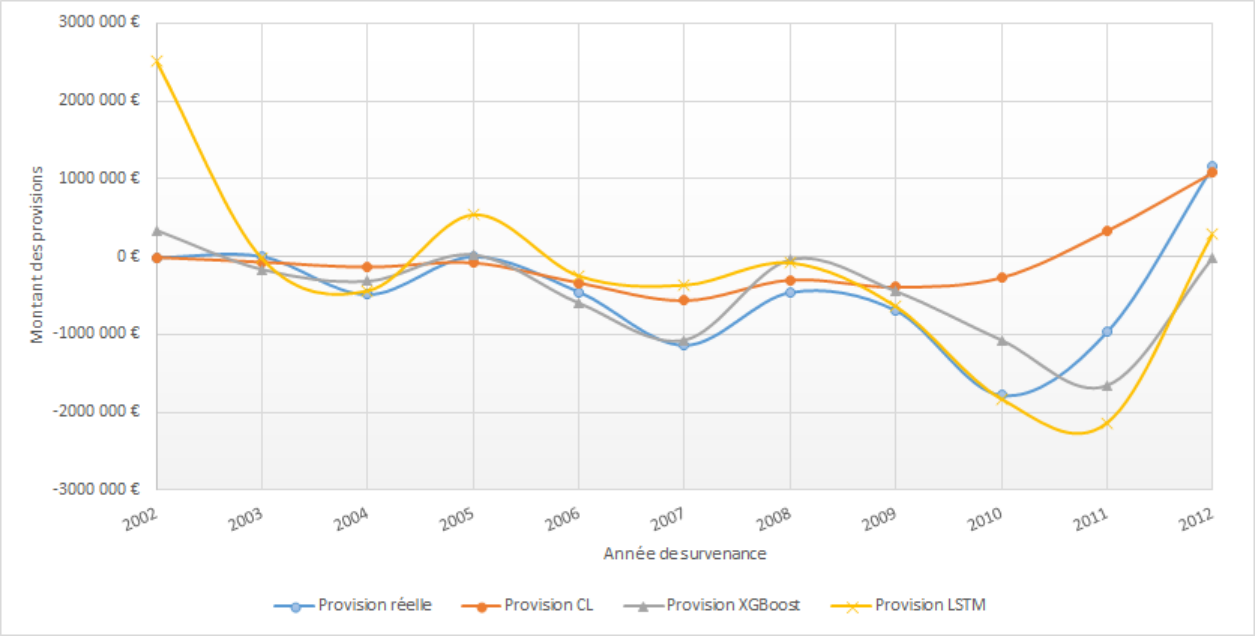


FIGURE 5 – Comparaison de l'estimation des charges totales avec Chain-Ladder, XGBoost et LSTM

Jusqu'en 2009, même si la méthode de Chain-Ladder estime légèrement moins bien les tendances réelles que les méthodes XGBoost et LSTM, les quatre courbes restent pour autant assez proches.

La véritable différence entre les nouvelles techniques et Chain-Ladder est visible sur les dates de survenance les plus récentes (i.e. à partir de 2010) où les écarts entre l'approche classique et les provisions observées explosent. Les méthodes XGBoost de LSTM s'en sortent mieux en collant plus fidèlement aux provisions réellement observées.

Le propos précédent est toutefois à nuancer car pour les sinistres avec une année de survenance en 2012 ou en 2002, c'est bien Chain-Ladder qui a l'estimation la plus proche de la réalité observée.

Contexte réglementaire

L'objectif de cette partie est de discuter de l'usage possible de nouvelles méthodes de provisionnement dans le cadre défini par la norme IFRS 17.

La norme IFRS 17 a pour but de résoudre les problèmes de comparabilité inhérents à la norme IFRS 4 qu'elle remplacera. Pour cela, elle impose à tous les contrats d'assurance d'être comptabilisés de manière identique quelque soit la localisation de la société ayant émis ce contrat. Cette uniformisation des rapports devraient bénéficier à la fois aux investisseurs et aux assureurs.

La norme IFRS 17, effective à partir de janvier 2021, impose un travail de segmentation sur les contrats d'assurance avant même de pouvoir les comptabiliser au bilan. Cette segmentation requiert le regroupement des contrats d'assurance en portefeuilles de risques homogènes, puis en cohorte annuelle. Le dernier échelon de segmentation repose sur un concept clé : les contrats onéreux. Les flux de trésorerie sortants pour ces types de contrat (en terme de charges ou de frais par exemple) sont supérieurs aux flux de trésorerie entrants. Et si un contrat est profitable au moment de son émission, il peut devenir onéreux à cause d'un sinistre déclaré par exemple. Ainsi, l'utilisation des ces nouvelles méthodes de provisionnement devraient permettre de déterminer si un contrat initialement profitable devient onéreux à la suite d'un ou plusieurs sinistres. Connaître ces catégories sera un enjeu car une fois déclaré comme onéreux, les pertes associés à un groupe de contrats sont immédiatement répercutées sur le résultat de l'entreprise.

Enfin, l'utilisation des estimations de charge ultime des algorithmes ne sera toutefois pas immédiate car il faudra retraiter tous les effets monétaires. De plus, il faudra être capable de représenter l'erreur commise par le modèle sous forme d'un intervalle de confiance.

Conclusion

L'utilisation de l'ensemble des données disponibles sur un sinistre apporte une véritable valeur ajoutée aux méthodes de provisionnement. De plus, ces approches semblent pouvoir s'inscrire dans le cadre défini par la future norme comptable concernant les contrats d'assurance : IFRS 17.

Il faut toutefois nuancer le propos précédent. Tout d'abord le bon usage d'approches de type apprentissage automatique requiert une grande qualité des données et un matériel informatique performant. De plus, les résultats d'une modélisation à l'ultime requièrent des retraitements avant d'être utilisables d'un point de vue comptable par exemple.

Executive Summary

Keywords : *Data Mining, Machine learning, IFRS, Non-life Insurance, Reserving, Automatic Learning, Neural network, Chain-Ladder, LSTM, XGBoost*

Introduction

Today, reserving, which is very often volatile, makes it difficult for companies to communicate about their results (either their financial results or the results linked to solvency indicators). Volatility can be explained in particular by arbitrations in the choice of reserving assumptions. At the same time, prudential regulations such as Solvency II require an ever more cautious management of commitments and an ever more transparent communication.

Nowadays, during the « digital era », pharaonic quantities of data [4] are recorded, whatever the field. The world of insurance does not escape this true digital revolution. In this recent context of massive information gathering, machine learning algorithms are supposed to be very useful tools adapted to the analysis of all these data. And insurance companies, with large amounts of data of all kinds, consider this revolution as a tremendous opportunity to exploit their online resources in order to improve risk management and establish a better strategy of reserving neither too optimistic (leading to undervaluation of the risk) nor too pessimistic (leading to unnecessary trappped capital).

Finally, besides the reliability of the calculation, the interest of the new methods is to be able to individualize the reserving. The purpose of this study is also to consider how it is possible to have for each disaster, a vision to the ultimate of what it will cost the company. This would be a step forward compared to traditional methods that work by aggregating lessons. And by aggregating, these traditional methods lose information. They do not take into account the heterogeneity of the development of the claims and do not use all the elements no doubt correlated to the evolution of the loss.

Thus, this thesis proposes a reminder of historical methods in a first part. These first results serve as a point of comparison for testing the performance of other methods. In a second part, two approaches are presented and used in line-by-line reserving. The first method is the XGBoost boosting gradient algorithm. The second approach is also derived from machine learning, the Long Short-Term Memory (abbreviated LSTM). Finally, in a third part, this study focuses on the possible contribution of these new approaches to the framework defined by IFRS 17.

Great care was first taken in the analysis and processing of the data base. The objective is to have a final version of each claim per quarter during the first five years of its development (ie 20 quarters). From this global base, a test base was constructed with claims occurring between 2002 and 2012 (in fact, the ultimate loss of these claims is, by hypothesis, known to June 2017, ie to today). The estimate of provisions made on this base will then be compared with the three methods presented below.

Finally, the starting point of the projection is the view at the end of June 2012 of the total loss of the test sample.

4. From a computer perspective, this is an elementary description, often coded, of a reality (chosen, fact, event ...)

The Chain-Ladder Method

The Chain-Ladder technique allows to project the amount of the claims loss using development factors. Claims are aggregated by year of occurrence.

First, the estimated reserve for claims with a year of occurrence between 2002 and 2012 is compared with the actual provisions observed, i.e. the amount of reserves that the insurance company needed to meet its commitments during the eleven fiscal years. This amount is observed today, namely June 2017.

Year of occurrence	Reserve observed	Estimated Reserve with Chain-Ladder
2002	-11 970 €	-8 487 €
2003	0 €	-69 830 €
2004	-488 232 €	-134 225 €
2005	-1 652 €	-79 148 €
2006	-454 292,00 €	-336 112 €
2007	-1 138 121 €	-564 787 €
2008	-461 045 €	-300 691 €
2009	-692 705 €	-392 928 €
2010	-1 782 887 €	-270 554 €
2011	-969 772 €	330 231 €
2012	1 154 722 €	1 077 049 €
Total	**-4 845 953 €**	**-749 482 €**

in euros

FIGURE 6 – Detailed analysis of reserves estimated by Chain-Ladder with actual values

It is found that in reality the loss of the claims in the test base has decreased, resulting in amounts of negative reserve.

On the total amount of reserve, Chain-Ladder commits an error of more than 85%. The traditional method has a great deal of difficulty in estimating the most significant amounts of reserves, as it is in 2010 and 2011. Thus, even if its simplicity of implementation and its auditability make it a very practical tool, Chain-Ladder's method thus seems to show some difficulties in estimating the reserves since the claims do not all have a homogeneous development.

Estimated reserves using the XGBoost Method

The XGBoost algorithm is an algorithm that aggregates trees. At each iteration, the new tree learns from the error committed by the previous tree. Thus, even if each tree has a weak predictive power, the decision rule constructed by summing the result of each tree is very reliable. In the context of our problem, this kind of algorithm has the advantage of being able to use all the information available on a claim.

The loss of each claim was projected to the ultimate using this algorithm. It was then possible to deduct the reserve estimated by the approach by comparing the amount of the projected loss to

the ultimate and the amount of the loss at the time of the projection. The results of these estimates are compared with those of Chain-Ladder (presented above) and the actual reserves observed.

Year of occurrence	Reserve observed	Estimated Reserve with Chain-Ladder	Estimated Reserve with XGBoost
2002	-11 970 €	-8 487 €	339 870 €
2003	0 €	-69 830 €	-164 520 €
2004	-488 232 €	-134 225 €	-314 675 €
2005	-1 652 €	-79 148 €	24 773 €
2006	-454 292,00 €	-336 112 €	-596 011 €
2007	-1 138 121 €	-564 787 €	-1 071 289 €
2008	-461 045 €	-300 691 €	-39 057 €
2009	-692 705 €	-392 928 €	-445 495 €
2010	-1 782 887 €	-270 554 €	-1 073 863 €
2011	-969 772 €	330 231 €	-1 655 465 €
2012	1 154 722 €	1 077 049 €	-15 950 €
Total	**-4 845 953 €**	**-749 482 €**	**-5 011 681 €**

in euros

FIGURE 7 – Comparison of actual and estimated loss by Chain-Ladder and XGBoost

Projection using XGBoost drastically reduces the overall error, which is now only 3% whereas it was 85% for Chain-Ladder.

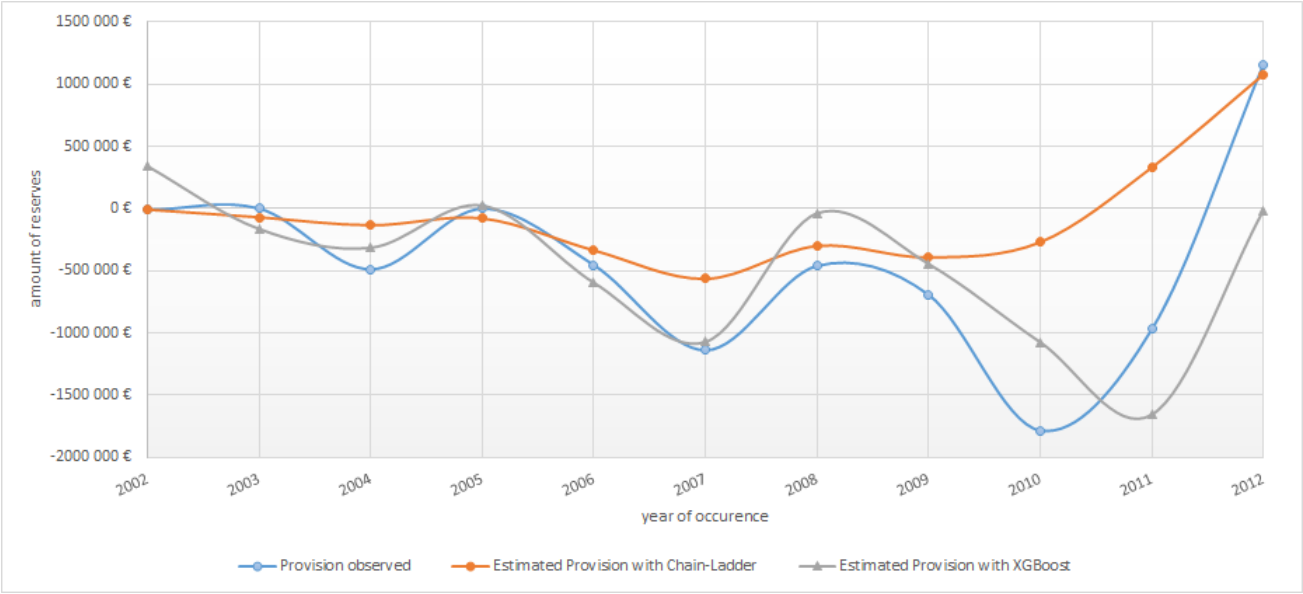


FIGURE 8 – Comparison of the estimate of reserves with Chain-Ladder and XGBoost by year of occurrence

Graphically we see that until 2009 the curves are very close. However, over the most recent years, except for 2012, the use of the XGBoost algorithm makes it possible to have provision estimates much closer to the amount of reserves observed than those of Chain-Ladder.

Finally, the XGBoost algorithm gives the possibility to visualize the variables it has used most to construct its final decision rule. In the cases of this study, among the three most significant

variables are the variables corresponding to the loss : the *reserve* and the *sum of the accumulated payments*.

Estimated reserves using the Long Short-Term Memory Method

The Long Short-Term Memory algorithm (abbreviated LSTM) belongs to the family of neural networks. Its specificity is to possess a memory. Indeed, thanks to a system of "gates" it is able to read, write or erase the information that was provided to it during its different iterations. The use of this approach is relevant in our case since it is considered that the amount of the ultimate loss of a claim is related to the different events that took place during its life.

Thus, as for the two previous approaches, the difference between the reserves actually observed and the estimated reserves for the years 2002 to 2012 was calculated.

Year of occurrence	Reserve observed	Estimated Reserve with Chain-Ladder	Estimated Reserve with XGBoost	Estimated Reserve with LSTM
2002	-11 970 €	-8 487 €	339 870 €	2 507 240 €
2003	0 €	-69 830 €	-164 520 €	-26 563 €
2004	-488 232 €	-134 225 €	-314 675 €	-439 624 €
2005	-1 652 €	-79 148 €	24 773 €	538 649 €
2006	-454 292,00 €	-336 112 €	-596 011 €	-248 235 €
2007	-1 138 121 €	-564 787 €	-1 071 289 €	-366 879 €
2008	-461 045 €	-300 691 €	-39 057 €	-79 211 €
2009	-692 705 €	-392 928 €	-445 495 €	-641 277 €
2010	-1 782 887 €	-270 554 €	-1 073 863 €	-1 829 730 €
2011	-969 772 €	330 231 €	-1 655 465 €	-2 134 382 €
2012	1 154 722 €	1 077 049 €	-15 950 €	294 528 €
Total	**-4 845 953 €**	**-749 482 €**	**-5 011 681 €**	**-2 425 484 €**

in euros

FIGURE 9 – Analysis of reserve estimation error with Chain-Ladder, XGBoost and LSTM

The XGBoost tree-based approach is the most accurate estimate of the overall reserve. Indeed, it is the closest to the actual overall reserve observed a posteriori. The error committed by XGBoost is a little more than 3 % (about 300 000 euros). Then, it is the LSTM algorithm that is closest to the observed value with a gap of more than 2.4 million euros. It is therefore the traditional approach by Chain-Ladder that is the furthest from the overall result (more than 4 million euros difference).

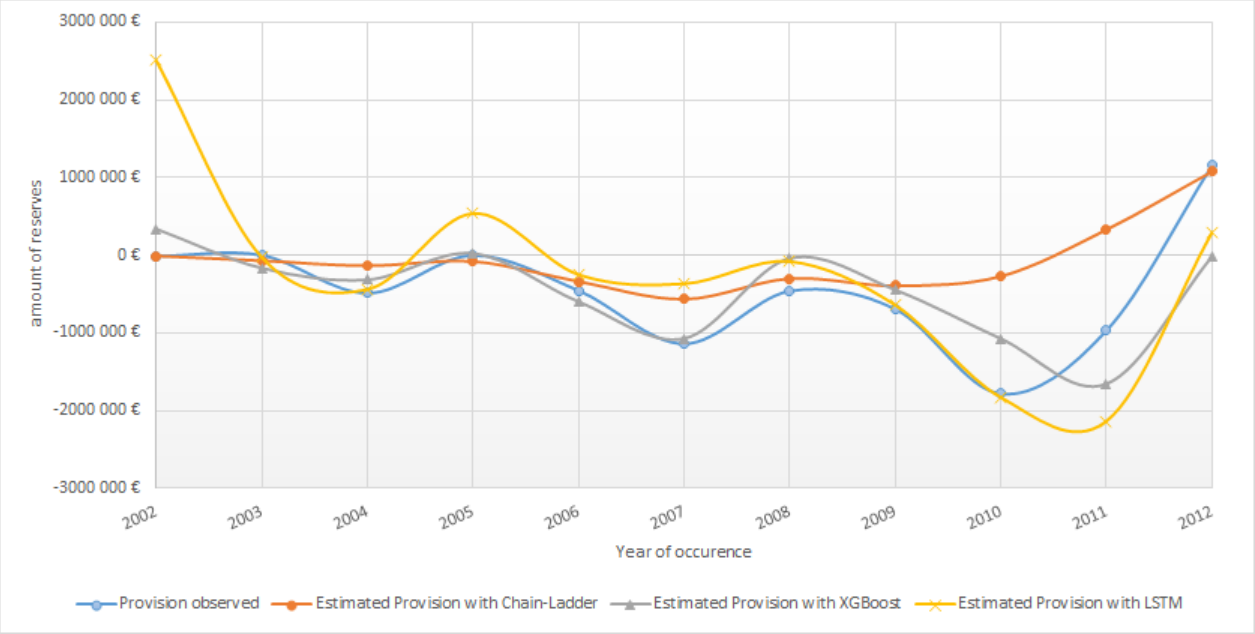


FIGURE 10 – Comparison of estimate of total loss with Chain-Ladder, XGBoost and LSTM

Until 2009, although the Chain-Ladder method is slightly less sensitive to real trends than the XGBoost and LSTM methods, the four curves remain fairly close. The true difference between the new techniques and the Chain-Ladder is visible on the most recent dates (i.e. from 2010) where the discrepancies between the classical approach and the observed ones explode. The XGBoost methods of LSTM do better by sticking more closely to the reserves actually observed.

However, the preceding statement is to be shaded because for claims with a year of occurrence in 2012, it is Chain-Ladder which has the most estimate close to the observed reality.

Regulatory Context

The objective of this section is to discuss the possible use of new methods of reserving within the framework defined by IFRS 17.

The purpose of IFRS 17 is to resolve the comparability problems inherent to IFRS 4 it will replace. To this end, it requires all insurance contracts to be accounted for in the same way regardless of the location of the company that issued the contract. This uniformity of reporting should benefit both investors and insurers.

The IFRS 17 standard, effective from January 2021, requires segmentation work on insurance contracts even before they can be recognized on the balance sheet. This segmentation calls for the consolidation of insurance contracts into homogeneous risk portfolios, followed by annual consolidation. The final level of segmentation is based on a key concept : onerous contracts. Out-of-pocket cash flows for these types of contracts (in terms of expenses or expenses, for example) are greater than the cash inflows. And if a contract is profitable at the time of its issue, it can become expensive because of a reported loss for example. Thus, using these new reserving methods should help determine whether an initially profitable contract becomes onerous as a result of one or more

claims. Knowing these categories will be an issue because once they are declared as onerous, the losses associated with a group of contracts are immediately passed on to the company's result.

Finally, the use of the ultimate loss estimates of the algorithms will not be immediate, however. It will be necessary to restate all the monetary effects. Moreover, it will be necessary to be able to represent the error committed by the model in the form of a confidence interval.

Conclusion

The use of all the data available on a claim brings real added value to the reserving methods. In addition, these approaches appear to be possible valuable tools to assist in the implementation of the future accounting standard for insurance contracts : IFRS 17

However, the preceding statement should be shaded. First, the proper use of machine learning approaches requires high data quality and high-performance computing hardware. In addition, the results of an ultimate model require restatements before being usable from an accounting point of view, for example.

Remerciements

Je remercie en premier lieu Christian KLUMPP mon tuteur de stage au sein de PwC. Je le remercie pour sa gentillesse, son investissement et toutes les connaissances qu'il a su me transmettre concernant le domaine du provisionnement. Je remercie également Alexandre VEBER pour m'avoir fait profiter de ses connaissances très avancées notamment en apprentissage automatique, pour la disponibilité dont il a fait preuve tout au long du stage.

Je remercie tout particulièrement Emmanuel DUBREUIL, associé au service Actuariat de PwC, pour son intérêt concernant mon étude, ses nombreuses remarques et suggestions qui m'ont permis d'améliorer la qualité de ce mémoire. Je remercie également Olfa HAJ TAIEB, directrice au service Actuariat de PwC pour son enthousiasme concernant mon projet et pour ses nombreux éclairages notamment sur la norme IFRS 17.

Je remercie très sincèrement Samia NOUAR avec laquelle j'ai travaillé sur cette étude pendant qu'elle préparait elle aussi son mémoire d'actuariat. Je la remercie de m'avoir fait profiter de son expérience, de son optimisme et de sa bonne humeur perpétuelle.

J'adresse également mes chaleureux remerciements à Jonathan KARSENTY et Aurélien PACARD pour leur aide précieuse tout au long de ce mémoire, en particulier sur des aspects opérationnels. Et de manière générale, je remercie l'ensemble des collaborateurs du service Actuariat de PwC pour l'intérêt qu'ils ont manifesté concernant mes travaux et leur bonne humeur quotidienne.

Je remercie également Vincent RIVOIRARD, mon tuteur pédagogique à l'université Paris dauphine pour avoir suivi mon travail sur ce mémoire et pour ses remarques pertinentes.

J'adresse mes sincères remerciements à l'ensemble des enseignants avec lesquels j'ai eu le plaisir d'étudier au cours de mon cursus à l'université Paris Dauphine. Je remercie enfin tous mes enseignants de l'INSA de Toulouse dont les enseignements m'ont été précieux au cours de mes travaux.

Table des matières

Introduction

Enjeux du mémoire

Aujourd'hui le provisionnement, très souvent volatile, rend difficile la communication des entreprises concernant leurs résultats (qu'ils soient financiers ou liés à des indicateurs de solvabilité). La volatilité peut s'expliquer notamment par des arbitrages à effectuer dans le choix des hypothèses de provisionnement. Dans le même temps, les réglementations prudentielles telles que Solvabilité 2 souhaitent une gestion des engagements toujours plus prudente et une communication toujours plus transparente.

À l'heure actuelle du « tout numérique », des quantités pharaoniques de données [5] sont recueillies, et ce, quel que soit le domaine. Le monde de l'assurance n'échappe pas à cette véritable révolution numérique. Dans ce contexte récent de collecte massive d'information, les algorithmes de machine learning s'avèrent être des outils très utiles et adaptés à l'analyse de toutes ces données. Et les assureurs, détenteurs de grandes quantités de données en tout genre, voient dans cette révolution une opportunité formidable d'exploiter leurs ressources en donnée afin d'améliorer la gestion de leurs risques et donc de mieux répondre aux exigences réglementaires. De même, l'utilisation de ces informations devrait permettre d'établir une meilleure stratégie de provisionnement ni trop optimiste (entraînant une sous-évaluation du risque) ni trop pessimiste (entraînant une immobilisation non nécessaire de capitaux). L'utilisation croissante du machine learning semble donc marquer le début d'une nouvelle ère dans le secteur de l'assurance.

Enfin, outre la fiabilité du calcul, l'intérêt des nouvelles méthodes est de pouvoir individualiser le provisionnement. Le but de ce mémoire est également de s'interroger sur la manière dont il est possible d'avoir pour chaque sinistre, une vision à l'ultime de ce qu'il coûtera à l'entreprise. Ce serait un pas en avant comparé aux méthodes traditionnelles qui fonctionnent en agrégeant les sinistres. Et en agrégeant, ces méthodes traditionnelles perdent de l'information. Elles ne prennent pas en compte l'hétérogénéité du développement des sinistres et n'utilisent pas non plus tous les éléments sans doute corrélés à l'évolution de la sinistralité.

Problématique

L'objectif de ce mémoire est de réfléchir à des approches utilisant le plus d'information possible sur les sinistres afin d'obtenir des estimations de provision moins volatiles et plus précises que les estimations faites avec des méthodes de provisionnement traditionnelles. En effet, les techniques telles que Chain-Ladder ou Bornhuetter-Ferguson ont montré qu'elles étaient de puissants outils quant à l'analyse du développement global de la charge de sinistres au fil des années. Cependant, un des reproches qu'il est possible de faire est la perte d'information qui est inhérente à ces méthodes. En agrégeant toute l'information sur les sinistres dans un triangle, une grande quantité d'information est perdue et n'est donc pas exploitée. Dans le même temps, le développement des technologies et l'accès à de nouvelles données semblent être des pistes prometteuses pour améliorer ces approches classiques.

5. D'un point de vue informatique, il s'agit une description élémentaire, souvent codée, d'une réalité (chose, fait, événement...)

Afin de mieux comprendre l'idée, il est intéressant de prendre l'exemple de l'analyse d'une image. Supposons qu'étant en vacances une personne prenne une image en haute résolution. Pour des raisons de stockage, cette personne est obligée de compresser sa photo en un format bien plus réduit. De retour chez elle, elle essaie alors de raconter la scène en se basant sur la photo compressée. Dans cette situation, bien que les éléments principaux puissent être assez simplement retrouvés, à savoir que l'image (Figure 11, à gauche) montre une fontaine avec différents personnages, il est difficile de déterminer si les formes sont des femmes ou des hommes ou même les inscriptions sur les montants. Il faudrait pour cela revenir à l'image originale (Figure 12, à droite). C'est ce problème que l'on retrouve avec les triangles de développement en essayant de retrouver les éléments de l'image globale (à savoir la charge de sinistre à l'ultime) en se basant sur une version compressée de l'information (réduite à un triangle).

FIGURE 11 – Image compressée

FIGURE 12 – Image originale

La fontaine de Trévi (Italie, Rome) en basse résolution (à gauche) et haute résolution (à droite).

Une autre façon de considérer le problème est d'imaginer qu'une compagnie d'assurance ait 1000 valeurs de la charge de sinistres d'une branche dont la longueur est de 5 ans. Autrement dit, les 1000 valeurs s'inscrivent dans un triangle avec 5 années d'occurrence et 5 années de développement. Ces montants de charge proviennent du suivi du développement des différents sinistres. Si cette entreprise calcule de manière classique ses provisions à l'aide d'un triangle de développement alors elle n'utilisera pas toutes les valeurs de charge dont elle dispose. En effet, pour la première année de survenance des sinistres (année $N-4$, N étant l'année actuelle), elle utilise 5 valeurs qui correspondent aux montants de la charge de sinistres à chaque fin d'année de développement. Pour l'année suivante, soit N-3, l'entreprise utilisera 4 valeurs pour les 4 années de développement dont elle dispose. Ainsi, pour construire son triangle de charges, la société d'assurance utilise 5+4+3+2+1 soit 15 valeurs de montants de charges de sinistres pour réaliser sa projection. Sur les 1000 valeurs dont elle disposait donc initialement, l'entreprise laisse donc de côté $\frac{1000-15}{1000} = \frac{985}{1000} = 98,5\%$ de l'information qu'elle possédait. Ce taux de perte d'information ne prend pas en compte d'éventuelles données dont disposerait l'entreprise sur ses clients ou sur les sinistres considérés.

L'autre enjeu lié à ces nouvelles technologies est la réduction de l'arbitrage et donc une probable réduction du risque opérationnel.

C'est pour essayer de remédier à ce genre de problématiques que ce mémoire s'interroge sur l'utilisation de méthodes plus modernes pour répondre aux enjeux de provisionnement.

Ainsi, ce mémoire propose un rappel sur les méthodes historiques dans une première partie. Ces premiers résultats servant de point de comparaison pour tester la performance des autres méthodes. Dans une deuxième partie, deux approches sont présentées et utilisées dans le cadre du provisionnement ligne à ligne. La première méthode est l'algorithme de gradient boosting XGBoost. La seconde approche est également issue du machine learning, il s'agit du Long Short Term Memory (abrégé LSTM). L'intérêt de ces approches est de pouvoir individualiser le calcul des provisions en incorporant toutes les informations pertinentes à la disposition de l'assureur. Enfin, dans une troisième partie, ce mémoire s'intéresse au possible apport de ces nouvelles approches au cadre défini par IFRS 17.

Chapitre 1

Contexte du mémoire

1.1 Rappel des enjeux

L'objectif de ce mémoire est de présenter deux approches innovantes qui ont pour but d'améliorer les méthodes de provisionnement classiques. Ces nouvelles approches doivent permettre de mieux prendre en compte l'information disponible. Pour cela, il est intéressant de penser à une granularité différente dans les pas de temps autre que celle en années.

Ces nouvelles techniques de calcul des provisions devraient permettre d'avoir des résultats plus fiables, moins marqués par le jugement d'expert d'une part. D'autre part, elles devraient permettre l'individualisation et donc le suivi à une échelle très fine du coût de chaque sinistre.

L'objectif de cette étude est de prévoir le coût à ultime de chaque sinistre, et ce, à n'importe quelle moment de sa vie. Cela implique de savoir où se situe dans le temps la notion d'ultime. En d'autres termes, il faut savoir quelle est l'horizon temporel au-delà duquel on se projette. Une autre approche possible est de considérer une approche par cadence. Cela consiste à déterminer les facteurs de développement de la charge de sinistralité jusqu'à l'ultime. Le tableau suivant présente quelques avantages et inconvénients de chaque modélisation.

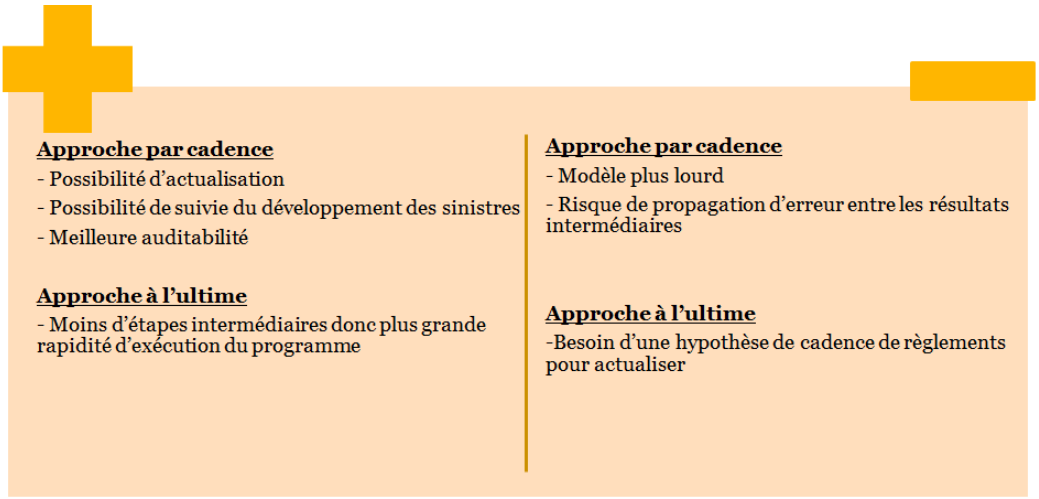


FIGURE 1.1 – Avantages et inconvénients de l'approche par cadence et de l'approche directement à l'ultime

La première approche évoquée dans cette étude est une méthode d'apprentissage supervisée basée sur les arbres de décision. Il s'agit de l'algorithme XGBoost.

La seconde technique employée repose, elle, sur les réseaux de neurones. Il s'agit du Long Short Term Memory (ou encore LSTM).

Enfin, il est très important de garder à l'esprit que nous cherchons à résoudre des problématiques concrètes. Ainsi, l'évaluation des algorithmes dépendra également de leur simplicité de compréhension, de leur rapidité d'exécution entre autres. Les performances mathématiques seules ne suffiront pas dans le cadre de ce mémoire. Les réflexions, avantages et inconvénients des différentes approches ainsi que la méthodologie de calcul seront aussi présentés.

1.2 Vie d'un sinistre

Pour rappel, un sinistre est un évènement entraînant des dommages ou des pertes susceptibles d'être indemnisés. Ces évènements, avec leur probabilité d'occurrence, constituent ce que l'on appelle des risques. L'assureur s'engage à supporter le risque dont veut se débarrasser un assuré en contrepartie du paiement d'une prime.

Ci-après est décrit le schéma de vie d'un sinistre quelconque :

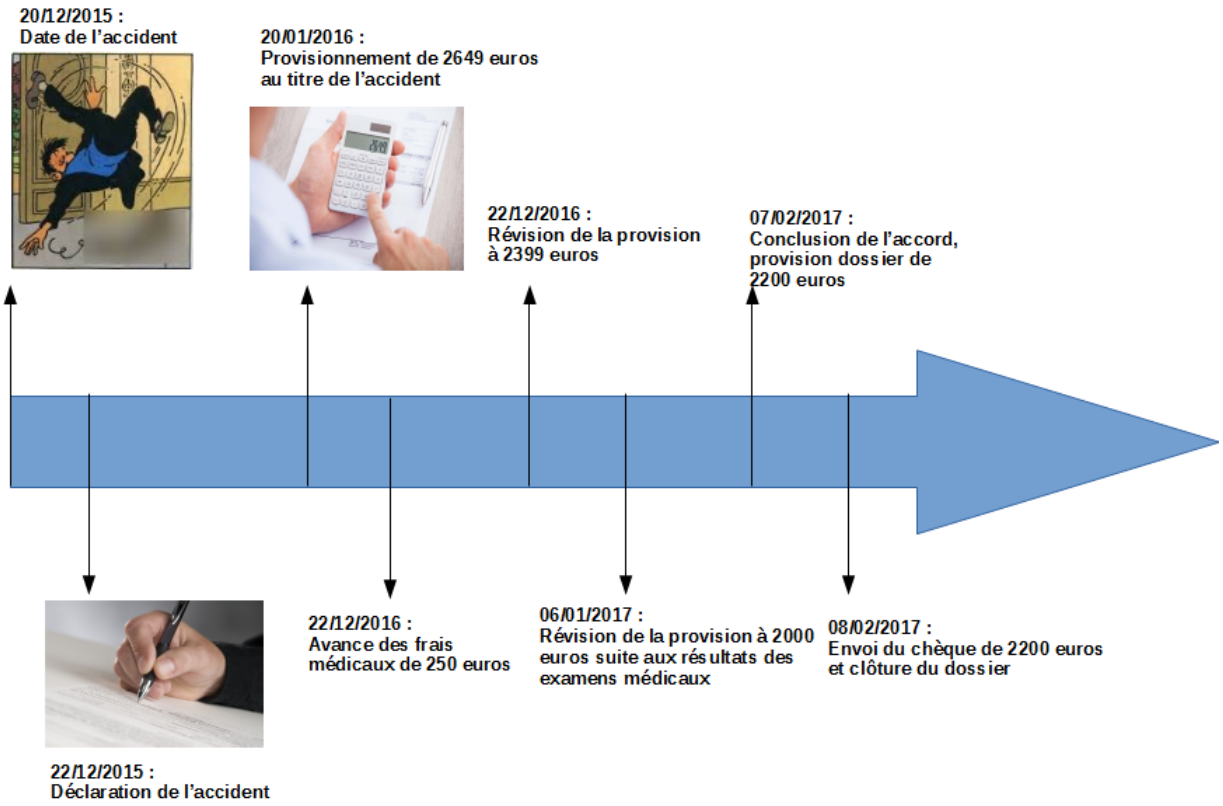


FIGURE 1.2 – Schéma de la vie d'un sinistre

Ainsi, comme présenté sur le schéma ci-dessus, entre le montant estimé à l'ouverture du sinistre et le chèque envoyé pour le clore, il s'écoule un laps de temps plus ou moins important.

Cet écart a un impact sur la gestion de la trésorerie de l'entreprise car l'entreprise doit toujours avoir suffisamment de provisions pour faire face à ses engagements. Elle a donc besoin de savoir le montant qu'elle devra payer pour un sinistre et quand est-ce qu'elle devra le faire. C'est le travail des équipes de provisionnement (ou "reserving" en anglais).

1.3 Étude de la base de données

1.3.1 Présentation générale

Le jeu de données utilisé provient d'un bancassureur français. La base est constituée de 9 tables qui peuvent être regroupées en trois grandes catégories : les tables liées aux victimes, celles liées aux sinistres et enfin celles liées aux contrats. C'est au total près de 800 00 contrats et 500 000 sinistres qui ont été mis à la disposition de cette étude. Le tableau ci-après résume le nombre de contrats et de sinistres disponibles par table.

Nom de la table	Nombre de contrats	Nombre de sinistres
Table 1	848 882	
Table 2	848 882	
Table 3	848 882	
Table 4	346 227	547 164
Table 5	68 402	86 836
Table 6		76 206
Table 7		86 836
Table 8		76 206
Table 9		112 460

FIGURE 1.3 – Descriptif des contrats et sinistres contenus dans chaque table

Ainsi, certaines tables ne contiennent que des informations sur les contrats (telles que les tables 1, 2 et 3). D'autres ont des informations relatives à la fois aux contrats et aux sinistres. Enfin, d'autres tables contiennent uniquement des informations sur les sinistres. Comme présenté en introduction, il est important de disposer du plus d'information possible concernant les sinistres. Et, c'est en croisant les données provenant de ces différentes tables que l'on disposera de la vue la plus précise possible de la situation d'un sinistre à un instant donnée.

Malgré cette profusion d'information, seules les données sur les sinistres de la branche Garantie contre les accidents de la Vie (appelée GAV par la suite) sont considérées dans le cadre de ce mémoire. En effet, pour une première approche, il était important d'avoir une branche avec un historique important et une certaine fiabilité dans les données. Ainsi, les données conservées concernent la GAV pour les années de survenance de 1982 à nos jours. Dans le même temps, les autres données non directement liées aux sinistres telles que la nature des échanges entre le bancassureur et l'assuré ou encore le nombre de bénéficiaires de la police sont conservés.

Pour rappel, dans l'administration française[1], la GAV est définie de la façon suivante : « La garantie contre les accidents de la vie (GAV) vise à protéger l'assuré, et éventuellement sa famille, des conséquences des accidents de la vie quotidienne. L'assureur indemnise la victime si le responsable de l'accident n'est pas identifié ou s'il en est lui-même le responsable. »

Enfin, puisque le nombre de sinistres ayant eu lieu avant 2002 est faible (on n'en compte que 6) et compte tenu des possibles effets monétaires et temporels difficiles à évaluer sur une si large période, seules les données depuis 2002 ont été conservées. Le périmètre de la modélisation s'étend

1. cf. site service-public.fr

donc entre 2002 et 2017. En considérant les premiers éléments évoqués ci-avant, le nombre des sinistres est alors réduit à 76 206.

Analyse de la variable cible

Parmi les bases disponibles, l'une fournie la vision actuelle du montant des charges auxquelles doit faire face le bancassureur. La société d'assurance définit le montant de la garantie totale à un instant donnée de la manière suivante :

Charge totale = Montant des provisions restantes + Montant total des paiements - Montant des prévisions de recours restantes - Montant total des recours encaissés

Dans la pratique, une fois que l'on dispose d'une estimation de la charge totale à l'ultime, il est possible de déduire la provision à constituer en calculant l'écart entre la valeur de charge actuelle et celle estimée à l'ultime.

La dernière vision de ces charges est résumée dans le tableau ci-après :

Nombre de sinistres	Moyenne	Ecart-type	Min	Quantile 25%	Quantile 50%	Quantile 75%	Max
76 206	2 231,19	21 369,42	- 8 167,00	0,00	0,00	0,00	1 113 760,00

FIGURE 1.4 – Statistique du montant des charges au mois de juillet 2017

Certains éléments sont remarquables dans le tableau ci-dessus. En premier lieu, on peut remarquer qu'il existe des montants de charges négatives. Ce cas est rare et concerne 19 sinistres sur les 76 206. Et, ces valeurs s'expliquent par le fait qu'au moment de l'extraction des données, le montant des recours (en prévision ou encaissés) était supérieur au montant des charges supportées par l'entreprise.

Le second point remarquable est lié à la distribution des données. En effet, on remarque que les quantiles à 25%, 50% et 75% sont tous nuls. Cela traduit une part très importante de charges ayant un montant nul dans le portefeuille du bancassureur. Ce fait est encore plus frappant lorsque l'on regarde la distribution du montant des charges totales.

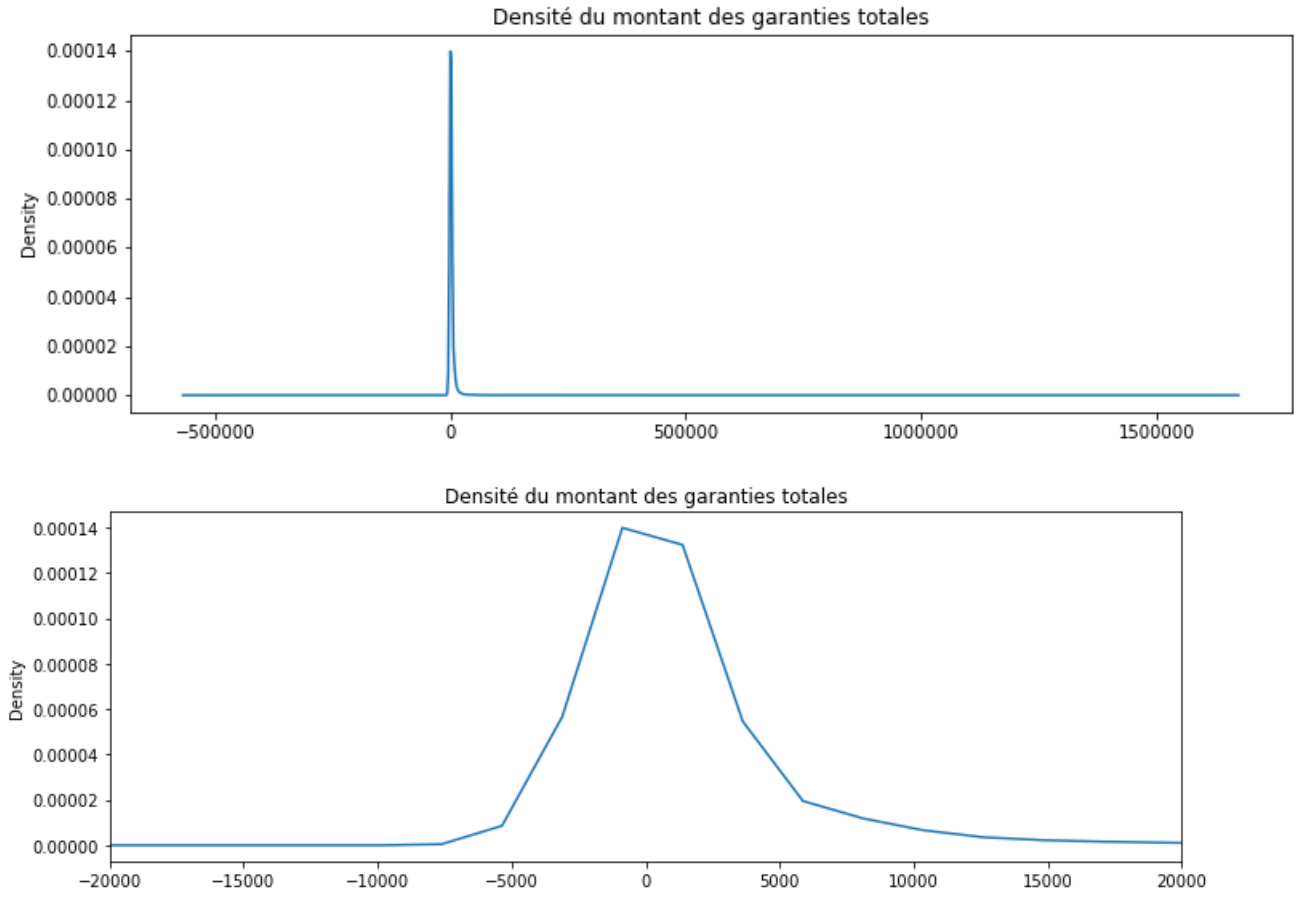


FIGURE 1.5 – Densité empirique du montant des garanties totales

Il est important de rappeler que le montant des garanties est compris entre -8 167 et 1 113 760 euros. Les valeurs au-delà de cette fenêtre (graphique du haut) sont une extrapolation de la densité des données. En outre, on remarque un pic de la densité autour de la valeur 0. Cette particularité s'explique par une politique de l'entreprise consistant à ouvrir systématiquement un sinistre dès qu'une déclaration est effectuée par une personne. Par la suite, ce sont près de 80% de ces sinistres ouverts qui sont déclarés sans suite. Cette constatation est importante car elle aura un impact non négligeable sur la stratégie de modélisation.

Une autre hypothèse est le fait que le modèle devrait être capable de gérer dans le même temps les sinistres graves et attritionnels. Pour cette raison, aucune modélisation particulière ne sera effectuée sur les sinistres graves en particulier.

Ces constatations particulières faites, on remarque aussi que le montant de garantie moyen est de 2 231 euros environ. De plus, le montant global des garanties au moment de l'extraction était de 171 143 359.27 euros. Il s'agit de la vue à aujourd'hui de la dernière diagonale du triangle constitué par les sinistres de la base de donnée.

Un des aspects importants de cette étude a été le travail fait pour construire une base de travail unique à partir des différentes bases fournies. Ce traitement a été effectué essentiellement à l'aide du logiciel informatique RStudio.

1.3.2 Prise en compte du temps

Le temps est un élément important de la modélisation. Premièrement, parce qu'il détermine la finesse avec laquelle l'étude est réalisée (le pas de temps pouvant être journalier tout aussi bien qu'annuel). L'autre aspect dans lequel il intervient est l'horizon de projection. En effet, en fonction de cette période supposée ultime, des impacts peuvent intervenir et nécessiter un retraitement (un impact monétaire avec besoin d'actualisation par exemple). Cette notion est primordiale car c'est elle qui détermine la structure même de la base utilisée pour la modélisation. Du temps dépendra le nombre de sinistres utilisables pour la modélisation, du temps dépendra également le nombre de versions dont disposera chaque sinitre dans la base.

Ainsi, au lieu d'agréger l'information à un niveau annuel, il paraît intéressant de commencer par une approche trimestrielle. Cela semble être pertinent lorsque l'on sait que de nombreuses entreprises sont soumises à la publication de rapports réglementaires trimestriels. C'est par exemple le cas des organismes devant répondre aux exigences du Pilier 3 de la directive Solvabilité 2. Ces entreprises sont amenées à recalculer tous les trois mois leurs provisions. De ce point de vue, le choix d'un pas de temps trimestriel semble cohérent. Toutefois, il est tout à fait possible d'envisager une autre valeur pour ce paramètre, i.e. de le choisir journalier ou semestriel par exemple.

Concernant, l'horizon de l'ultime, le tableau suivant a été un élément de décision important.

État	Quantile 90% durée vie*	Quantile 99,5% durée vie*	Durée vie moyenne*
SANS SUITE	2,00	5,20	1,10
NON GARANTI	4,30	7,90	2,20
FERMÉ	4,40	9,60	2,20

** en années*

FIGURE 1.6 – Éléments statistiques sur la durée de vie des sinistres

Les sinistres sans suite ou non garantis ont une durée de vie plus courte que les sinistres fermés normalement ce qui est cohérent.

Ce que l'on peut constater ensuite est qu'il existe un écart important entre les quantiles à 90% et à 99,5% de la durée de vie des sinistres fermés : plus de 5 ans. Il faut également garder en tête que le pas de temps étant trimestriel, chaque année de développement supplémentaire rajoute quatre périodes de développement en plus. De plus, pour pouvoir calibrer le modèle, il faut que les sinistres considérés soient fermés ou aient une durée de vie supérieure ou égale à l'horizon de temps choisi. Ainsi, plus celui-ci est grand et moins le nombre de sinistres utiles à l'apprentissage est important. Il y a donc un arbitrage à faire. Or on constate que la durée de vie moyenne des sinistres est plutôt faible (à peine plus de 2 ans). Et, près de 9 sinistres sur 10 se sont fermés au bout d'un peu moins de 4 ans et demi. Ainsi, choisir un horizon de temps de 5 ans semble être un bon compromis entre un nombre important de sinistres clos (plus de 90%), une contrainte sur la durée de vie et un développement en trimestres raisonnables. C'est donc cet horizon qui sera utilisé.

De manière pratique, cela signifie que les sinistres ayant une date survenance après le 12 décembre 2012 ne pourront pas être utilisés pour l'apprentissage. C'est une contrainte forte qui a de nombreuses conséquences. Par exemple, il manquera pour les prédictions sur ces sinistres cinq années d'inflation. Il sera donc nécessaire d'effectuer un retraitement de l'estimation de charge obtenue.

1.3.3 Base de travail finale

Il a fallu structurer la base en ayant une version par trimestre de chaque sinistre. Les bases avec les flux financiers, les échanges avec l'assuré et le statut du sinistre possédaient un historique. Il a donc était plus simple de récupérer une valeur trimestrielle pour ces informations. Pour récupérer ces valeurs historiques avec un pas de temps trimestrielle, il a été nécessaire d'agréger l'information à un niveau journalier. Différents évènements associés à un même sinistre pouvaient avoir lieu au cours de la même journée, et de ce fait, plusieurs lignes étaient créées le même jour pour un même sinistre. Le problème engendré était l'impossibilité d'effectuer des jointures car trop de doublons étaient générés. Le premier travail a donc consisté à trouver un moyen d'agréger la base en minimisant la perte d'information. Pour cela, de nombreux arbitrages ont dû être réalisés. Il fallait par exemple déterminer si certaines variables peu renseignées devaient être conservées ou non dans la base. Il s'agissait encore de déterminer comment traiter les variables qualitatives : fallait-il compter le nombre d'occurrence de la modalité dans la journée ou simplement conserver l'indication que la modalité était apparu (en créant une variable booléenne par exemple). Ces choix ont sans aucun doute eu un impact sur la qualité de la modélisation qui a suivi. C'est sans doute ici, une première limite de l'approche. D'autant que cette nécessité d'agréger l'information est de nouveau apparu lorsque pour un même contrat par exemple, tous les bénéficiaires d'un même avenant étaient listés. La jointure devenait une nouvelle fois irréalisable et un travail de synthèse de la donnée fut également nécessaire.

Toutefois, toutes les variables n'avaient pas nécessairement d'historique. Parfois, il existait juste la valeur au moment de l'extraction des données. Ce problème d'historique existait pour les variables issues des bases autres que celles contenant les flux financiers, les échanges avec l'assuré et le statut des sinistres.

n° sinistre	Trimestre de développement	Flux financiers	Echanges avec l'assuré	Statut du sinistre	Autres variables
		☑ Historisé			☒ Non historisé

FIGURE 1.7 – Schéma de l'historique de la base de données

Pour palier à ce problème, il a été construit un historique basé sur la table contenant les échanges avec l'assuré. L'hypothèse étant que les différentes interactions entre l'assureur et l'assuré contenaient (au travers des courriers, des visites d'experts par exemple) toute l'information liée au sinistre. Ainsi, le premier travail a consisté à réduire le nombre de modalités de la variable contenant la nature des demandes. De 162 modalités initiales, la réduction a conduit à regrouper les modalités en 36 grandes catégories. Puis, il a fallu effectuer pour chaque modalité le décompte cumulé du nombre d'échanges à chaque trimestre pour chacun des sinistres.

n° sinistre	Trimestre de développement	Interaction Avocat	Interaction Expert	Interaction Medicale
S1	0	0	0	0
S1	3	0	0	1
S1	6	0	1	3
S1	9	1	1	4
S1	12	2	1	6
S1	15	2	1	6

FIGURE 1.8 – Schéma du décompte des échanges par nature d'échange et par trimestre

Enfin, un « mapping », une correspondance a été établie entre les 36 nouvelles modalités de la variable nature des échanges et les variables non historisées. L'hypothèse ayant servi à construire la base est que la valeur des variables non historisées est connue uniquement à partir du trimestre de développement où tous les échanges de la modalité de référence sont observés.

Supposons par exemple que le sinistre S_1 ait un taux IPP de 95 et soit lié à la modalité *Interaction Medicale*. Alors, on supposera que l'information sur le taux IPP est connue à partir du trimestre 12 (car tous les échanges liés à l'interaction médicale sont connues à partir du trimestre 12). Le schéma suivant résume l'exemple décrit juste avant.

n° sinistre	Trimestre de développement	Interaction Avocat	Interaction Expert	Interaction Medicale	Taux IPP
S1	0	0	0	0	0
S1	3	0	0	1	0
S1	6	0	1	3	0
S1	9	1	1	4	0
S1	12	2	1	6	95
S1	15	2	1	6	95

FIGURE 1.9 – Exemple d'historisation du taux IPP

Enfin, dans le cadre de ce mémoire, il est considéré qu'au bout de 5 ans (ou 20 trimestres), la plupart des sinistres ont atteint, ou sont tout proche de leur développement ultime. Ainsi, pour chaque sinistre, seuls les 20 premiers pas de développement ont été conservés. L'idée est déterminer si l'algorithme est capable d'évaluer la bonne valeur de charge totale ultime dès les tous premiers développements du sinistre.

En définitive, c'est une base de 6851 sinistres et de 530 variables qui a servi à l'étude. Parmi ces variables se trouvent de nombreuses données sur les sinistres qui ne sont pas utilisées par les méthodes classiques.

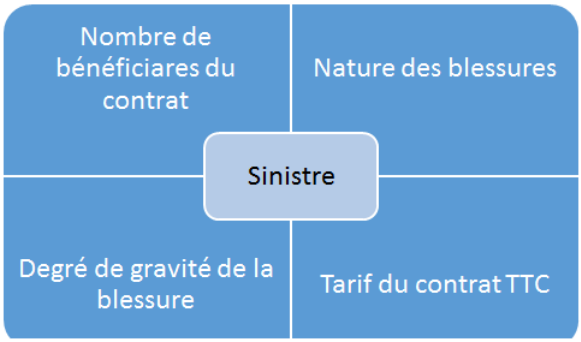


FIGURE 1.10 – Exemple de variables de la base de données

Ce sont les variables de la matrice précédente ainsi que bien d'autres qui seront fournies en entrée à l'algorithme. Le but restant de déterminer si ces nouvelles informations permettent une meilleure prédiction des garanties totales ultimes et a fortiori des provisions.

De cette base globale est créée deux échantillons :
— Une base d'apprentissage qui servira à calibrer le modèle
— Une base test qui servira à quantifier la performance du modèle

Un point de méthodologie

La variable à prédire correspond à la vue à juin 2017 des sinistres ayant une date de survenance antérieure à 2012. En d'autres termes, il s'agit du montant de la charge de sinistre à juin 2017 pour les sinistres avec une année de survenance avant 2012.

La base ne contient que les cinq premières années de la vie du sinistre. Ainsi en prenant comme variable cible la vue à aujourd'hui, un léger biais est introduit dans la modélisation. En effet, pour les sinistres les plus anciens (i.e. avec une année de survenance en 2002 ou 2003 par exemple), l'information sur la charge a pu changer entre la vue à cinq ans et la vue à aujourd'hui. De fait, pour les sinistres plus anciens, il existe un léger biais entre la charge prédite et l'information utilisée pour la prédiction. Le choix de l'horizon de temps joue donc un rôle primordial dans la modélisation. Toutefois, l'hypothèse de travail étant que la charge d'un sinistre a pratiquement atteint son développement maximal au bout de 5 ans, la modélisation définie précédemment est cohérente.

Chapitre 2

Méthodes de provisionnement classiques

2.1 Méthode de Chain-Ladder

2.1.1 Présentation générale

La méthode de Chain-Ladder est une approche fréquemment utilisée depuis les années 1930. À chaque période de développement, un facteur de développement est calculé, permettant de passer d'une période à l'autre. Cette technique aurait été développée par Thomas F. Tarbel [1] dans la revue « Casualty Actuarial Society ». La méthode s'inspirant elle-même, sans doute, de calculs ou méthodes utilisés par des souscripteurs ou des comptables. Cependant, de part sa simplicité de compréhension et sa simplicité opérationnelle, cette méthode déterministe est devenue une véritable référence. Cette approche pouvant en plus être employée sur différents types de triangles, qu'il s'agisse de la charge ou du nombre de sinistres par exemple. Dans ce mémoire, la technique est appliquée sur des triangles de charges cumulées.

Soit N$\in \mathbb{N}$ un nombre d'années.
Soit également le facteur de développement $f_{i,j}$ défini pour $i + j \leqslant$ N par :

$$f_{i,j} = \frac{S_{i,j+1}}{S_{i,j}}$$

ce qui est équivalent à :

$$f_{i,j} \times S_{i,j} = S_{i,j+1}$$

où $S_{i,j}$ est la charge de sinistres cumulée des sinistres avec une année de survenance i et une année de développement j. Les valeurs i et j sont homogènes à des années.

Avec cette seconde forme, la notion de "chaîne" est plus facilement compréhensible. Les valeurs de charges sont en effet chaînées grâce aux facteurs de développement.

1. Tarbell, T. F. "Incurred But Not Reported Claim Reserves," Proceedings of the Casualty Actuarial Society 20, 1933—1934, pp. 275—280

2.1.2 Hypothèse fondamentale de Chain-Ladder

Pour tout $j \in [\![0\ ;\ N-1]\!]$, les coefficients de passage $f_{i,j}$ sont indépendants de l'année d'occurrence du sinistre i. On peut alors poser f_j comme étant le coefficient commun à l'année de développement j.

Reformulé autrement, la méthode de Chain-Ladder considère que les années d'occurrence sont indépendantes.

Des facteurs f_j calculés précédemment, il est possible de déduire la charge de sinistres à l'ultime :
- $\forall i \in [0, n]$,

$$C_i = S_{i,n-i} \prod_{k=n-i}^{n-1} f_k$$

On peut alors calculer le montant des provisions à constituer :
- $\forall i \in [0, n]$

$$R_i = C_i - S_{i,n-i}$$

Enfin, des réserves estimées pour chaque année de survenance, on calcule le montant global à provisionner :

$$R = \sum_{i=0}^{n} R_i$$

2.1.3 Validation du modèle

Rappelons que pour toute année de développement j fixé, on peut trouver un facteur f_j tel que :
- $\forall i \in [0, n-j-1] : S_{i,j+1} = S_{i,j} * f_j$ (I)

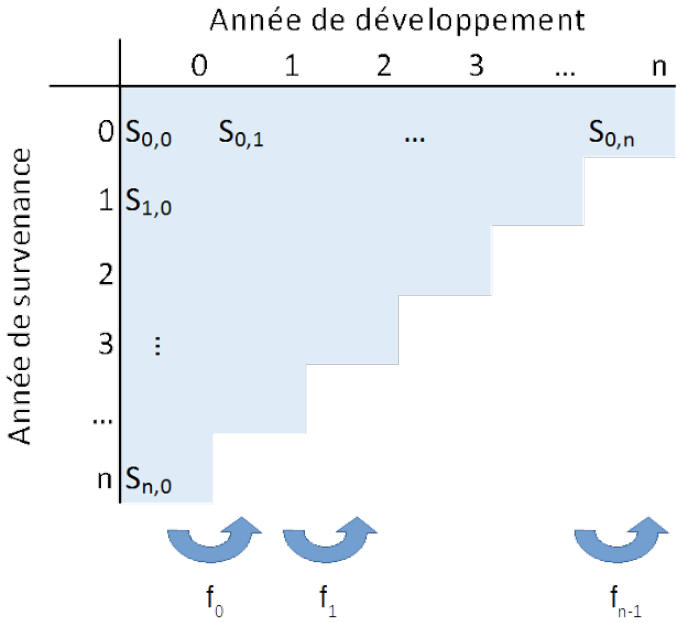


FIGURE 2.1 – Principe de développement des cadences par Chain-Ladder

L'égalité (I) implique donc que les (n-j)-couples $(S_{i,j}, S_{i,j+1})$ (i.e. les charges de sinistre pour deux années de développement consécutives) doivent être relativement alignés par rapport à une droite passant par l'origine. En d'autres termes, les éléments d'une même colonne j doivent tous se développer à la période suivante à l'aide du même coefficient multiplicatif f_j. Si ce n'est pas le cas, une analyse doit être effectuée afin de choisir des facteurs de développement f_j plus adaptés. C'est d'autant plus important si sont présents dans le triangle de développement des valeurs extrêmes ou des comportements non aléatoires et donc non représentatifs du comportement global des sinistres.

De plus, si les années de survenance sont indépendantes, alors on devrait observer une certaine constance dans les coefficients individuels d'une même année de survenance.

2.2 Méthode de Bornhuetter-Ferguson

L'approche de Bornhuetter-ferguson est aussi basée sur des triangles de développement. Cependant, alors que la méthode de Chain-Ladder estime les provisions en se basant sur le comportement des sinistres d'une année sur l'autre, Bornhutter-Ferguson estime la charge à l'ultime et l'évolution de la charge des sinistres connus actuellement vers cette charge ultime.

2.2.1 Définition

Les cadences de développement sont définis par $\gamma_k = \frac{E[S_{i,k}]}{E[S_{i,N}]}$ pour $k \in \{1, \dots, N\}$. Il s'agit du pourcentage de règlement effectué au bout des k premières années de développement.
Ainsi, on a que : $\gamma_N = 1$

Les facteurs de développement sont définis par : $\lambda_k = \frac{E[S_{i,k+1}]}{E[S_{i,k}]}$. De manière très pratique, on ne peut déterminer les cadences de développement qu'à partir de la première année de d'occurrence car c'est la seule pour laquelle on connaît le déroulement des règlements à 100%.
De plus on a : $\gamma_k = \prod_{i=k}^{N-1} \frac{1}{\lambda_i}$.

2.2.2 Modélisation

La méthode de Bornhuetter-Ferguson repose sur l'idée que l'on possède des vecteurs $\alpha = (\alpha_1, \dots, \alpha_N)$ et $\gamma = (\gamma_1, \dots, \gamma_N)$ avec $\gamma_N = 1$ tels que :

$\forall i, k = 1, \dots, N \ E[S_{i,k}] = \gamma_k \alpha_i$.

On a donc : $\alpha_i = E[S_{i,N}]$ et $\gamma_k = \frac{E[S_{i,k}]}{E[S_{i,N}]}$. C'est ce que l'on définit comme la cadence de développement.
Comme évoqué précédemment, les γ_k peuvent être estimés en déroulant la cadence de la première année de développement.

Basé sur les éléments décrits ci-avant, on a alors la relation suivante :

- $E[S_{i,k}] = E[S_{i,N+1-i}] + (\frac{E[S_{i,k}]}{E[S_{i,N}]} - \frac{E[S_{i,N+1-i}]}{E[S_{i,N}]})E[S_{i,N}]$
- $E[S_{i,k}] = E[S_{i,N+1-i}] + (\gamma_k - \gamma_{N+1-i})E[S_{i,N}]$

De là, on obtient les estimateurs de Bornhuetter-Ferguson suivants :
$\hat{S}_{i,k} = S_{i,N+1-i} + (\hat{\gamma}_k - \hat{\gamma}_{N+1-i})\hat{\alpha}_i$,
où $S_{i,N+1-i}$ est connu car se trouvant sur la diagonale.

À partir de la modélisation initiale on peut calculer la provision globale de Bornhuetter-Ferguson ($\hat{R}^{BF}$) :

$\hat{R}^{BF} = \sum_{i=1}^{N} S_{i,N} - S_{i,N+1-i} = \sum_{i=1}^{N}(1 - \hat{\gamma}_{N+1-i})\hat{\alpha}_i$

où $(1 - \hat{\gamma}_{N+1-i})\hat{\alpha}_i$ est l'estimation par Bornhuetter-Ferguson de la provision pour l'année d'origine i.

2.3 Application des méthodes agrégées

Sauf mention contraire explicite, le triangle présenté le long de cette section est constitué des sinistres qui serviront à calibrer le modèle.

	0	3	6	9	12	15	18	21	24	27	30	...
31/03/2002	0,00	40 362,00	40 362,00	46 263,10	64 727,10	61 390,80	66 987,67	57 427,58	57 427,58	52 273,58	52 273,58	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
30/06/2013	1 296 643,03	1 836 356,40	1 949 272,30	2 053 287,48	2 329 684,64	2 558 232,84	2 714 912,17	2 662 952,39	2 702 510,55	2 726 603,26	2 880 923,26	
30/09/2013	1 351 774,06	2 968 130,00	3 124 938,50	4 124 108,69	4 410 038,53	6 060 136,25	6 047 820,71	6 231 916,03	5 675 138,96	5 728 907,07	5 774 786,48	
31/12/2013	896 697,46	2 091 690,20	3 371 406,30	3 805 554,22	4 256 296,28	4 330 740,87	3 433 163,81	3 412 932,29	3 780 996,19	3 959 003,58	4 182 987,49	
31/03/2014	1 453 408,28	2 274 688,80	2 738 376,50	3 145 788,42	3 225 355,74	3 269 162,98	3 385 611,32	3 425 051,46	3 556 519,02	3 592 572,68	3 639 830,38	
30/06/2014	2 526 726,53	3 343 314,90	4 495 245,30	4 554 273,55	4 661 572,13	4 705 046,13	4 413 913,77	4 606 038,27	4 828 169,58	4 805 849,41	4 743 544,61	
30/09/2014	2 982 769,30	5 574 236,40	6 022 125,50	6 098 597,19	6 077 231,28	6 041 937,52	6 001 639,32	6 011 336,87	6 005 993,61	6 020 047,78	6 013 029,35	
31/12/2014	2 589 180,28	3 452 143,70	4 397 609,40	4 413 528,93	5 377 745,02	5 348 817,42	5 282 835,67	5 321 984,22	4 922 209,98	4 939 449,75	5 169 250,12	
31/03/2015	3 273 602,65	3 864 983,50	4 099 103,40	4 687 676,14	4 806 629,76	4 859 411,94	4 877 439,04	4 860 677,57	4 967 273,56	5 149 354,35	-	
30/06/2015	3 457 377,06	4 331 295,50	4 588 474,10	4 724 303,96	4 807 441,22	4 828 348,15	4 926 125,22	5 372 929,18	5 437 806,22	-	-	
30/09/2015	4 200 234,45	5 454 513,10	5 458 498,90	5 570 801,35	5 682 933,70	5 965 157,08	5 957 421,72	5 952 116,80	-	-	-	
31/12/2015	3 804 784,37	4 236 553,10	4 952 162,90	5 046 891,82	5 168 977,36	5 202 264,72	5 346 852,36	-	-	-	-	
31/03/2016	2 748 858,19	3 772 952,30	4 046 878,70	4 179 958,65	4 265 807,37	4 405 929,44	-	-	-	-	-	
30/06/2016	2 945 822,47	3 874 948,50	4 295 783,40	4 367 264,34	4 753 955,52	-	-	-	-	-	-	
30/09/2016	3 212 433,56	5 489 809,10	5 622 323,60	5 709 712,38	-	-	-	-	-	-	-	
31/12/2016	3 104 704,90	4 068 818,90	4 508 672,80	-	-	-	-	-	-	-	-	
31/03/2017	2 706 875,95	4 211 714,90	-	-	-	-	-	-	-	-	-	

FIGURE 2.2 – Triangle tronqué des charges totales cumulées

2.3.1 Application de la méthode de Chain-Ladder

Vérification des hypothèses

Pour rappel, l'hypothèse fondamentale de Chain-Ladder considère que les années d'occurrence des sinistres sont indépendantes. Sous cette hypothèse, on doit alors avoir que :

- $\forall j \in [0, N-1]$,

$$\frac{S_{0,j+1}}{S_{0,j}} = \frac{S_{1,j+1}}{S_{1,j}} = ... = \frac{S_{N-j-1,j+1}}{S_{N-j-1,j}}$$

Ci-après sont présentés les nuages de points du couple $(S_{i,j}, S_{i,j+1})$ pour $j = 3$ et $j = 6$ et $\forall i \in [0, N-j-1]$.

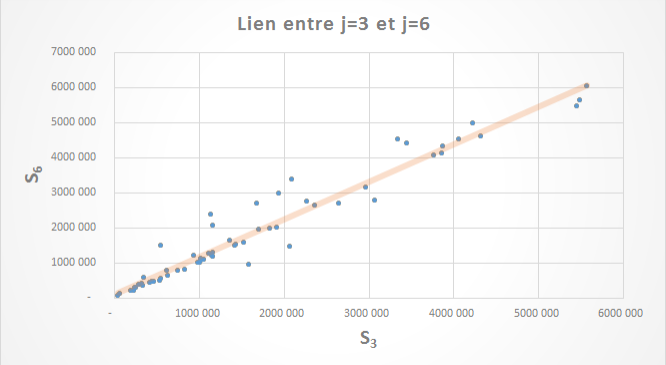


FIGURE 2.3 – Analyse de la linéarité du couple (S_3, S_6)

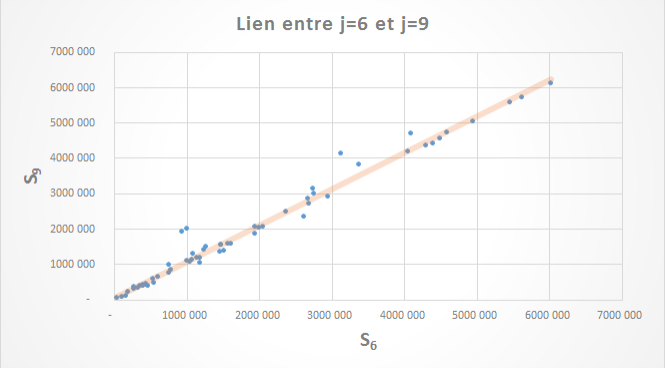


FIGURE 2.4 – Analyse de la linéarité du couple (S_6, S_9)

Les droites en orange représentent les tendances linéaires associées à chaque nuage de points. Comme on peut le constater, ces droites représentent de manière fiable le comportement des couples (S_3, S_6) et (S_6, S_9). Et, en admettant que ces observations pour les deux premiers couples soient vraies pour le reste du triangle, on a donc que l'hypothèse d'indépendance des années d'occurrence est vérifiée. Il existerait donc des facteurs de développement.

Le triangle de développement est défini par les facteurs de développement $f_{i,j}$. Et, pour que l'hypothèse fondamentale soit vérifiée, il est nécessaire que pour $j \in [0, N-1]$, les $f_{i,j}$ soient relativement constants. Les graphiques graphiques suivants représentent les facteurs de développement individuels pour $j = 3$ et $j = 6$.

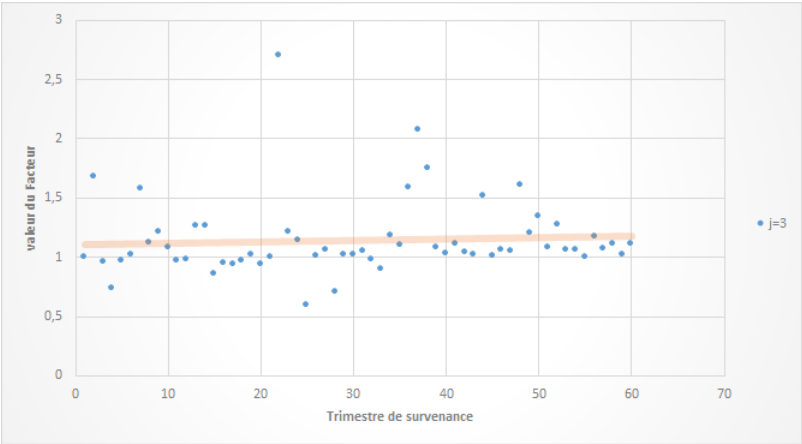


FIGURE 2.5 – Analyse de la constance des facteurs de développement pour $j = 3$

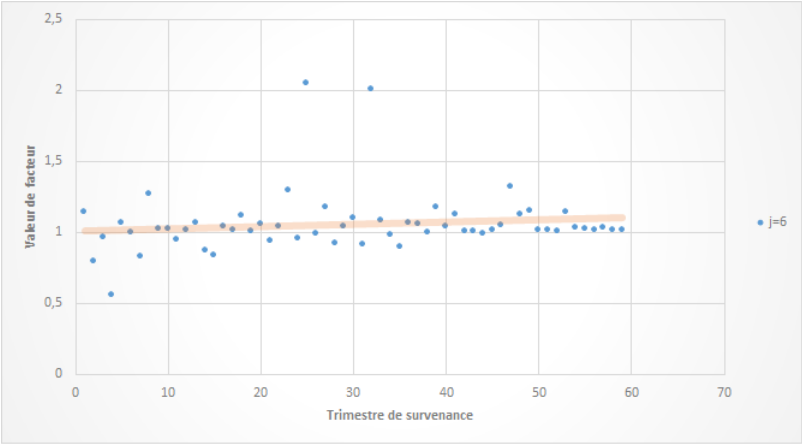


FIGURE 2.6 – Analyse de la constance des facteurs de développement pour $j = 6$

On constate qu'il existe quelques individus atypiques. Cependant, de manière globale, on retrouve bien que les facteurs sont regroupés autour d'une certaine droite. On peut donc admettre, que l'hypothèse d'indépendance des facteurs de développement est avérée.

Application aux triangles de charges totales

Une fois la méthode de Chain-Ladder utilisée, on obtient les facteurs de développement. Cependant, l'étude de ces facteurs révèlent parfois la présence de comportements atypiques. Ce fut le cas pour le facteur associé au trimestre de développement 84.

Trimestre de survenance	31/03/2002	30/06/2002	30/09/2002	31/12/2002	31/03/2003	30/06/2003	30/09/2003	31/12/2003	31/03/2004	30/06/2004	30/09/2004
Valeur du facteur	1,00	1,00	1,00	1,00	1,00	1,00	0,82	1,00	1,00	1,00	1,00

Trimestre de survenance	31/12/2004	31/03/2005	30/06/2005	30/09/2005	31/12/2005	31/03/2006	30/06/2006	30/09/2006	31/12/2006	31/03/2007	30/06/2007
Valeur du facteur	0,99	1,00	1,00	1,00	1,00	1,00	1,00	0,45	1,00	1,00	1,00

Trimestre de survenance	30/09/2007	31/12/2007	31/03/2008	30/06/2008	30/09/2008	31/12/2008	31/03/2009	30/06/2009	30/09/2009	31/12/2009	31/03/2010
Valeur du facteur	1,00	1,00	1,00	0,77	1,00	1,00	1,00	1,00	1,00	1,00	1,00

FIGURE 2.7 – Facteurs de Chain-Ladder individuels pour le trimestre de développement 84

La valeur du facteur agrégée pour cette période est 0.94. Le nombre de 84 mois correspond à une durée de développement de 7 ans. Il est donc surprenant d'avoir un coefficient si bas alors que le déroulement du sinistre est plutôt bien avancé et alors que les facteurs précédents et suivants sont eux autour de 1.

Grâce au tableau précédent qui présente les facteurs individuels de Chain-Ladder pour les pas de temps 84, on constate que sur les 33 valeurs, trois se différencient fortement des autres. Il s'agit des facteurs en orange pour les trimestres de survenance 30/09/2003 et 30/09/2006 et du facteur de développement pour le trimestre de survenance 30/06/2008. Ces facteurs sont anormalement bas et expliquent en grande partie le niveau du facteur agrégé alors que ces trois éléments sont marginaux. Une étude de la base révèle alors que chacun des facteurs de la période est expliqué majoritairement par la variation d'un seul sinistre entre les deux périodes. Ainsi, pour le pas de temps 84, un sinistre est passé d'une charge totale cumulée au temps 84 de 920 217 euros à une charge totale cumulée de 3253 euros. Il y a donc eu une baisse de la charge de 900 000 euros. Cette variation s'explique par la clôture du sinistre au cours de ce trimestre. Ainsi, la charge totale cumulée finale correspond au montant totale des paiements. Et, les montants payés furent bien inférieurs aux provisions prévues pour le sinistre. Une explication similaire est valable pour les deux autres facteurs marginaux.

Ces problèmes sont très visibles chez les sinistres les plus anciens de la base. Afin de palier à ce genre de contraintes, il paraissait cohérent de considérer des facteurs de Chain-Ladder basés sur des sinistres plus récents. Ainsi, seuls les facteurs basés sur les 7 années de sinistralité les plus récentes seront considérés dans le calcul des différents facteurs. Les nouveaux coefficients de Chain-Ladder sont présentés dans le tableau suivant.

Période de développement	0	3	6	9	12	15	18	21	24	27	30	33	36
Valeur du facteur	1,73	1,19	1,08	1,06	1,05	1,02	1,02	1,02	1,04	1,03	1,01	1,01	1,00

Période de développement	39	42	45	48	51	54	57	60	63	66	69	72	75
Valeur du facteur	0,97	0,99	0,99	0,99	0,99	0,99	1,00	1,00	1,00	1,00	1,00	0,95	1,00

Période de développement	78	81	84	87	90	93	96	99	102	105	108	111	114
Valeur du facteur	0,99	0,98	0,96	0,98	0,98	0,99	1,00	0,98	1,05	0,99	0,98	0,97	0,99

Période de développement	117	120	123	126	129	132	135	138	141	144	147	150	153
Valeur du facteur	0,98	1,00	0,98	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00

Période de développement	156	159	162	165	168	171	174	177	180
Valeur du facteur	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00

FIGURE 2.8 – Facteurs de Chain-Ladder retraités

On peut constater qu'une plus grande homogénéité existe au sein des facteurs. Cependant, on peut considérer que la valeur de 0,95 pour le pas de temps 72 est également suspecte. Toutefois, les optimisations étant sans fins, ce sont les coefficients ci-présents qui seront conservés.

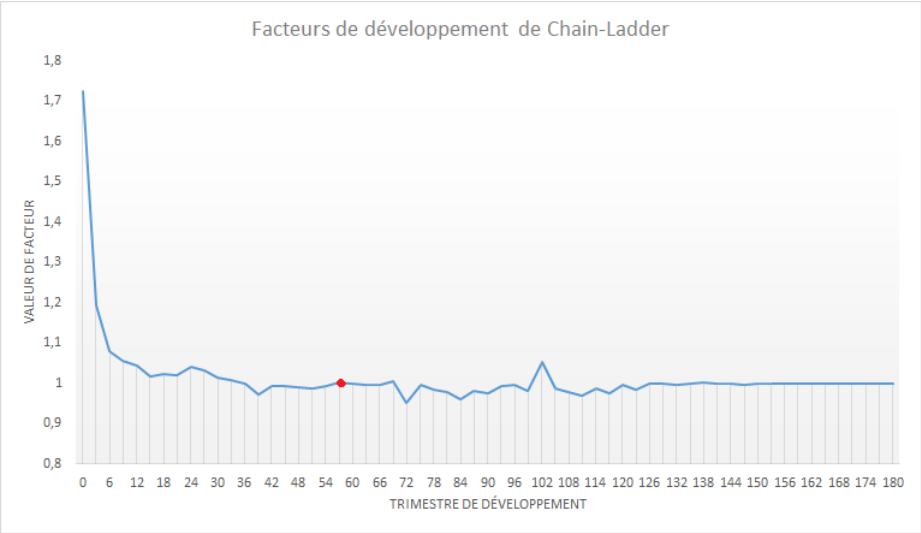


FIGURE 2.9 – Graphique des facteurs de développement obtenus par Chain-Ladder

Il apparaît qu'après le point orange, les coefficients sont stables autour de 1 malgré de légères oscillations au pas de temps 72 et 102. Or, ce point correspond au facteur obtenu après de 60 mois de développement, soit 5 ans. Or, 5 ans est l'horizon de temps choisi pour l'ultime. Cet élément conforte donc le choix de cet horizon de temps. Ainsi au bout de cinq années, la plupart des sinistres ont atteint plus de 99% de leur cadence de développement.

	0	3	6	9	12	15	18	21	24	27	30	...
31/03/2002	0,00	40 362,00	40 362,00	46 263,10	64 727,10	61 390,80	66 987,67	57 427,58	57 427,58	52 273,58	52 273,58	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
30/06/2013	1 296 643,03	1 836 356,40	1 949 272,30	2 053 287,48	2 329 684,64	2 558 232,84	2 714 912,17	2 662 952,39	2 702 510,55	2 726 603,26	2 880 923,26	
30/09/2013	1 351 774,06	2 968 130,00	3 124 938,50	4 124 108,69	4 410 038,53	6 060 136,25	6 047 820,71	6 231 916,03	5 675 138,96	5 728 907,07	5 774 786,48	
31/12/2013	896 697,46	2 091 690,20	3 371 406,30	3 805 554,22	4 256 296,28	4 330 740,87	3 433 163,81	3 412 932,29	3 780 996,19	3 959 003,58	4 182 987,49	
31/03/2014	1 453 408,28	2 274 688,80	2 738 376,50	3 145 788,42	3 225 355,74	3 269 162,98	3 385 611,32	3 425 051,46	3 556 519,02	3 592 572,68	3 639 830,38	
30/06/2014	2 526 726,53	3 343 314,90	4 495 245,30	4 554 273,55	4 661 572,13	4 705 046,13	4 413 913,77	4 606 038,27	4 828 169,58	4 805 849,41	4 743 544,61	
30/09/2014	2 982 769,30	5 574 236,40	6 022 125,50	6 098 597,19	6 077 231,28	6 041 937,52	6 001 639,32	6 011 336,87	6 005 993,61	6 020 047,78	6 013 029,35	
31/12/2014	2 589 180,28	3 452 143,70	4 397 609,40	4 413 528,93	5 377 745,02	5 348 817,42	5 282 835,67	5 321 984,22	4 922 209,98	4 939 449,75	5 169 250,12	
31/03/2015	3 273 602,65	3 864 983,50	4 099 103,40	4 687 676,14	4 806 629,76	4 859 411,94	4 877 439,04	4 860 677,57	4 967 273,56	5 149 354,35	5 326 445,17	
30/06/2015	3 457 377,06	4 331 295,50	4 588 474,10	4 724 303,96	4 807 441,22	4 828 348,15	4 926 125,22	5 372 929,18	5 437 806,22	5 661 024,34	5 855 126,46	
30/09/2015	4 200 234,45	5 454 513,10	5 458 498,90	5 570 801,35	5 682 933,70	5 965 157,08	5 957 421,72	5 952 116,80	6 056 716,50	6 305 340,46	6 521 534,53	
31/12/2015	3 804 784,37	4 236 553,10	4 952 162,90	5 046 891,82	5 168 977,36	5 202 264,72	5 346 852,36	5 403 618,61	5 498 579,26	5 724 292,08	5 920 563,48	
31/03/2016	2 748 858,19	3 772 952,30	4 046 878,70	4 179 958,65	4 265 807,37	4 405 929,44	4 542 181,91	4 590 405,17	4 671 074,79	4 862 819,13	5 029 552,82	
30/06/2016	2 945 822,47	3 874 948,50	4 295 783,40	4 367 264,34	4 753 955,52	4 831 226,51	4 980 631,21	5 033 509,38	5 121 965,91	5 332 219,01	5 515 047,23	
30/09/2016	3 212 433,56	5 489 809,10	5 622 323,60	5 709 712,38	5 905 483,89	6 001 471,87	6 187 066,17	6 252 752,77	6 362 635,71	6 623 817,43	6 850 931,27	
31/12/2016	3 104 704,90	4 068 818,90	4 508 672,80	4 888 975,77	5 056 606,31	5 138 796,60	5 297 712,85	5 353 957,39	5 448 045,32	5 671 683,76	5 866 151,36	
31/03/2017	2 706 875,95	4 211 714,90	4 895 646,48	5 308 590,37	5 490 608,43	5 579 853,01	5 752 408,83	5 813 480,78	5 915 644,16	6 158 477,19	6 369 635,70	

FIGURE 2.10 – Triangle tronqué des charges totales reconstituées

Application à l'échantillon test

L'échantillon test est le sous-groupe de sinistres qui permettra d'évaluer la qualité réelle de la prédiction de l'algorithme. En effet, ces sinistres n'ont pas servi durant la phase d'apprentissage ou de calibration du modèle. Ainsi, le résultat estimé ne sera plus biaisé par optimisme dans la mesure où les individus utilisés pendant l'apprentissage ne serviront pas à déterminer l'erreur commise par le modèle.

Dans la pratique, à l'aide des facteurs obtenus avec l'échantillon d'apprentissage, il est possible de développer la charge de sinistres des individus de l'échantillon test jusqu'à l'ultime. On obtient alors une charge totale de 20 150 123 euros.

Dans la réalité, la charge totale ultime observée pour l'échantillon test est de 16 053 652 euros. La méthode de Chain-Ladder a donc commis une erreur de plus de 4 millions d'euros. Ce qui est relativement conséquent.

Provision observée	Provision estimée avec Chain-Ladder
-4 845 953 €	-749 482 €

montants en euros

FIGURE 2.11 – Comparaison des provisions estimées par Chain-Ladder avec les valeurs réelles

Plus précisément, vue à fin 2012, la charge réelle était de 20 899 605 euros. Ainsi, les provisions exprimées dans le tableau précédent représentent le montant qu'il aurait fallu mettre en provision lorsque l'on était fin 2012 compte tenu de l'évolution de la charge réelle observée de ces sinistres à juin 2017.

On constate donc une diminution à posteriori de la charge totale de près de 5 millions d'euros. Cette diminution de la charge est liée à une stratégie de provisionnement trop pessimiste de la part de la compagnie. En ce qui concerne la méthode de Chain-Ladder, l'estimation est plus pessimiste que ce qui s'est produit dans la réalité.

En effet, l'approche par Chain-Ladder a prévu une baisse de la charge d'à peine plus de 1 million d'euros soit une erreur de 77% par rapport au montant de provision global à prévoir en réalité.

Année de survenance	Provision observée	Provision estimée avec Chain-Ladder
2002	-11 970 €	-8 487 €
2003	0 €	-69 830 €
2004	-488 232 €	-134 225 €
2005	-1 652 €	-79 148 €
2006	-454 292,00 €	-336 112 €
2007	-1 138 121 €	-564 787 €
2008	-461 045 €	-300 691 €
2009	-692 705 €	-392 928 €
2010	-1 782 887 €	-270 554 €
2011	-969 772 €	330 231 €
2012	1 154 722 €	1 077 049 €
Total	**-4 845 953 €**	**-749 482 €**

montants en euros

FIGURE 2.12 – Analyse détaillée des provisions estimées par Chain-Ladder avec les valeurs réelles

Les provisions observées et de Chain-Ladder correspondent aux montants vus à la fin du premier semestre 2017.

Lorsque l'on analyse année par année les provisions estimées par Chain-Ladder, on constate que l'approche traditionnelle respecte le comportement global réel, à savoir une diminution de la charge future sauf pour l'année de survenance 2012. Une erreur dans la tendance a tout de même eu lieu pour les sinistres ayant pour année de survenance 2011. La méthode de Chain-Ladder prévoit une hausse de la charge de 330 231 euros quand en réalité celle-ci diminue de pratiquement un million d'euros (969 772 exactement).

Toutefois, on observe que la technique a des difficultés à bien estimer les très hauts montants de provision (négatifs ou positifs). Ainsi, on constate une baisse réelle des charges de 1 138 121 euros en 2007 quand la méthode de Chain-Ladder ne prévoit qu'une baisse de 564 787 euros, soit une erreur de 49%. Ces écarts sur les montants les plus élevés de provision expliquent en grande partie l'écart global entre les provisions estimées par Chain-Ladder et les provisions réelles.

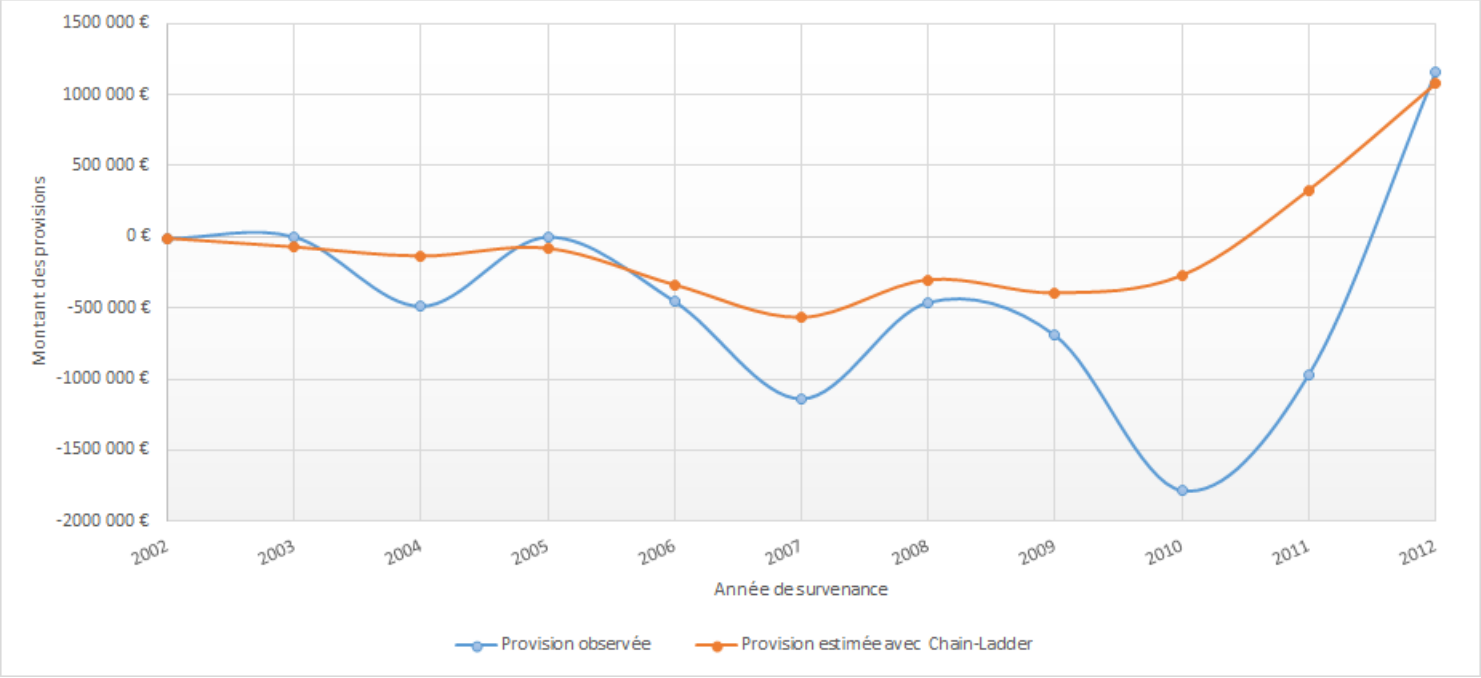


FIGURE 2.13 – Comparaison des charges totales par Chain-Ladder sur l'échantillon test

Le graphique ci-avant présente pour chaque année de survenance, le montant des provisions estimées par Chain-Ladder (en orange) à la provision réelle (en bleu).

On constate graphiquement que les points représentant les provisions sont très proches lorsque la valeur réelle de provision est inférieure à 500 000 euros (en valeur absolue). Pour les montants supérieurs à ce seuil l'écart est plus important.

Il faut également noter que les provisions réelles les plus importantes sont constituées pour les années de survenance les plus récentes. Cette constatation est traduite par le fait que les points en bleu situés après 2010 sont les plus éloignés de la droite des montants nuls. Ce constat semble cohérent dans la mesure où les sinistres survenus avant 2005 sont les plus anciens, il paraît donc raisonnable que les mouvements de charge soient plus faibles en ce qui les concerne.

Enfin, la méthode de Chain-Ladder semble avoir du mal à imiter la tendance globale des provisions réelles. En effet, alors que les points en bleu ont une tendance plutôt à la baisse (excepté le dernier point en 2012), les provisions estimées par Chain-Ladder semblent elles stagner voire augmenter.

Ces premiers résultats serviront de point de comparaison afin d'estimer la qualité des modèles présentés dans les sections suivantes.

Chapitre 3

Estimation des provisions à l'aide de méthodes d'apprentissage automatique

3.1 Description de l'algorithme du gradient

Cet algorithme d'optimisation étant utilisé par les deux approches présentées dans ce mémoire, il est intéressant que le lecteur ait en tête son fonctionnement.

L'algorithme de descente du gradient est un algorithme d'optimisation différentiable. Il a pour but de minimiser une fonction réelle différentiable définie sur un espace euclidien. Dans sa version la plus simple, celle utilisée par les deux nouvelles approches XGBoost et LSTM, l'algorithme ne permet de trouver qu'un point stationnaire (i.e., un point pour lequel le gradient de la fonction à minimiser est nul) d'un problème d'optimisation sans contrainte.

De manière plus formelle on a alors :

Soient E un espace hilbertien (avec un produit scalaire $< \cdot, \cdot >$, et une norme associée $|| \cdot ||$) et $x \in \mathbb{E} \to f(x) \in \mathbb{R}$ une fonction différentiable. Posons $f'(x)$ et $\nabla f(x)$ respectivement la dérivée et le gradient de f en x. On a alors pour tout $d \in \mathbb{E}$: $f'(x) \cdot d =< \nabla f(x), d >$.

L'algorithme du gradient : Soient un point initial $x_0 \in \mathbb{E}$ et un seuil de tolérance $\epsilon \geq 0$. L'algorithme du gradient définit une suite de valeurs x_1, x_2,...$\in \mathbb{E}$ jusqu'à ce qu'une des conditions d'arrêt soit activée. Le test d'arrêt peut porter sur le nombre d'itérations de l'algorithme ou sur le seuil de tolérance. Pour un point x_k, le point suivant x_{k+1} est défini comme suit :

1. **Simulation** : calcul de $\nabla f(x_k)$
2. **Test d'arrêt** : $||\nabla f(x_k)|| \leq \epsilon$, arrêt
3. **Calcul du pas** $\alpha_k > 0$ par une règle de recherche sur f en x le long de la direction $-\nabla f(x_k)$
4. **Nouvel itéré** : $x_{k+1} = x_k -_k \nabla f(x_k)$

Le coefficient α est appelé le taux d'apprentissage.

3.2 Algorithme XGBoost

L'algorithme XGBoost est le sigle issu de Extreme Gradient Boosting. Il s'agit d'une autre désignation de l'approche Gradient Boosting Machine qui a été introduite par Friedman en 1999 dans

«Greedy Function Approximation : A Gradient Boosting Machine »(Jerome H. Friedman [20]). L'algorithme XGBoost repose sur cette approche originale. Son implémentation a été initialement effectuée par Tianqi Chen avant la contribution de nombreux développeurs.

XGBoost est utilisé dans le cadre de problèmes d'apprentissage supervisé. Cela signifie que l'on utilise les données de calibrage (ou d'apprentissage) avec de nombreuses variables x_i pour prédire une variable cible $_i$. Cependant, à l'inverse de l'apprentissage non supervisé, on dispose alors d'une valeur observée y_i. Ainsi, on peut entre autre calculer l'erreur d'apprentissage du modèle (écart entre y_i et $\hat{y}_i$), que ce soit dans le cadre d'un problème de classification ou d'un problème de régression.

3.2.1 Lien avec notre problématique

Avantages de la méthode

L'algorithme XGBoost repose sur un formalisme mathématique poussé. C'est cette construction qui est à la base de ses performances.

D'un point de vue opérationnel, cette approche dispose d'un avantage majeur : comme tout algorithme construit à partir d'arbres de décisions, il est capable de restituer les variables les plus discriminantes du modèle qu'il a construit. C'est important dans la mesure où l'on cherche à estimer de manière plus fine les provisions d'une part, mais on cherche également à savoir parmi toutes les informations à la disposition de l'assureur, lesquelles ont le plus d'impact sur le montant du paiement à l'ultime. Avoir cette information est très utile pour deux raisons principales. La première est la volonté de pouvoir à terme réfléchir à un management action à l'échelle de la compagnie d'assurance qui lui permettrait de piloter ses provisions telle qu'elle pourrait piloter sa stratégie de tarification. Dans le contexte de concurrence très forte entre les assureurs, la moindre optimisation du résultat pourrait être déterminante. L'autre raison qui justifie l'importance de connaître les variables les plus significatives découle de contraintes réglementaires. En effet, la question du contrôle et de l'auditabilité de ces nouvelles méthodes jouera sans aucun doute un rôle important dans leur développement. Et, comprendre sur quelles bases l'algorithme a construit son résultat pourrait être une piste de réflexion intéressante.

Un autre argument motivant le choix de cet algorithme est sa capacité à prendre en compte l'interaction entre les différentes variables explicatives mises à sa disposition. En effet, parmi certaines variables qui seront fournies en entrée de l'algorithme, il peut exister des liens plus ou moins importants. Cette contrainte est très forte dans notre cas et nécessite un outil pouvant répondre à cette exigence. D'une manière plus globale, lorsque l'on traite une base de données contenant un très grand nombre de variables, il est important de comprendre comment celles-ci interagissent entre elles. En effet, ces similitudes ont un impact direct sur le choix du modèle et la qualité de la modélisation. Ainsi une modélisation à l'aide d'un modèle linéaire généralisé est techniquement réalisable tant qu'il n'existe pas de couples de variables liées à 100%. Pour autant, afin d'avoir un modèle de qualité, il est préférable de traiter ces liens en amont de la construction du modèle. En utilisant l'approche avec XGBoost, il est possible de s'affranchir du traitement préalable de ces similitudes même s'il est important de les avoir en tête.

Enfin, un autre avantage de cette technique est la possibilité de traiter efficacement des problèmes lorsque le lien entre la variable à prédire et les variables explicatives n'est pas linéaire. Cela permet d'envisager le traitement de problématiques complexes pour lesquelles il n'y a pas d'indices

à priori sur le lien entre le phénomène à expliquer et les variables dont on dispose pour modéliser le phénomène : comme la reconnaissance d'images ou dans notre cas la prédiction de la charge totale à l'ultime d'un sinistre.

Limites de la méthode

L'algorithme XGBoost est souvent qualifié de « boîte noire ». Une définition intéressante de cette expression est donnée par le site internet *Wikipédia* qui définit par boîte noire « tout système dont les entrées et sorties sont connues mais dont le fonctionnement interne reste peu ou pas du tout compris ». En ce sens, un cerveau est également une boite noire. Cette désignation de l'algorithme provient sans doute de deux mécanismes utilisés par l'approche.

Le premier mécanisme est le **Boosting**. Le boosting (Freund et Schapire [21]) est une branche de l'intelligence artificielle et regroupe de nombreux algorithmes. L'idée est d'améliorer la performance d'un faible classifieur, i.e., un modèle dont la qualité de prédiction est à peine supérieure à la moyenne. Ainsi, itérativement, l'information fournie par chaque faible classifieur est agrégée au classifieur final (ou strong classifieur). Le principe est qu'en regroupant de nombreuses règles de décisions peu efficaces on obtienne une règle finale très efficace.

Le second mécanisme impliqué est **l'algorithme du gradient**. Cet algorithme a été présenté dans la section précédente.

Enfin, outre les deux mécanismes précédents, l'approche possède aussi des paramètres liés à l'utilisation d'arbres comme classifieurs. Ainsi, l'algorithme XGBoost dépend d'un très grand nombre de paramètres.

3.2.2 Principe de l'algorithme XGBoost

Cette section a pour but de rendre un peu moins boîte noire l'algorithme XGBoost. La présentation ci-après est basée sur le travail initial des développeurs de l'algorithme (Tianqi Chen, Carlos Guestrin [19]).

Avant d'exposer plus en détails le fonctionnement de l'algorithme, il est sans doute utile d'effectuer quelques rappels concernant l'apprentissage supervisé.

Modèle et paramètres

Un **modèle** en apprentissage supervisé fait souvent référence à la structure mathématique qui permet d'effectuer une prédiction $\hat{y}_i$ à partir d'un élément x_i, pour $i \in [0; n]$ et n le nombre d'individus de la base de données. Ainsi, un modèle courant est le modèle linéaire où la prédiction est donnée par : $\hat{y}_i = \sum_j \hat{\theta}_j \times x_{i,j}$, une combinaison linéaire pondérée des j variables explicatives.

Les **paramètres** sont les inconnus que l'on cherche à estimer à partir des données. Dans le cas d'une régression linéaire par exemple, les paramètres sont les coefficients θ. Dans la suite de la section, θ fera référence aux paramètres du modèle.

La fonction objectif : fonction de perte et terme de régularisation

Quel que soit le problème que l'on souhaite résoudre à l'aide d'un modèle, on a toujours besoin d'un moyen de trouver une manière d'estimer les meilleurs paramètres pour le modèle. Pour parvenir à cela, il est nécessaire de faire appel à une fonction objectif. Cette fonction permet en effet de mesurer la performance d'un modèle en fonction d'un ensemble de paramètres.

Une fonction objectif peut être définie de la manière que l'on souhaite, pour autant, pour qu'elle soit pertinente, elle doit se composer d'une fonction de perte et d'un terme de régularisation :

$$Obj(\cdot) = L(\cdot) + K(\cdot) \tag{3.1}$$

où Obj est la fonction objectif composée par L, une fonction de perte et K un terme de régularisation.

La fonction de perte mesure la qualité de la prédiction du modèle sur les données fournies en apprentissage. Assez communément, pour un jeu de paramètres θ, la fonction de perte est l'erreur quadratique moyenne :

$$L(\theta) = \frac{1}{n}\sum_i (y_i - \hat{y}_i)^2 \tag{3.2}$$

Le terme de régularisation permet de contrôler la complexité du modèle et d'éviter le surapprentissage. Ce terme n'a pas de forme prédéfinie à priori et est souvent défini en fonction du problème. L'idée générale est qu'il doit permettre d'avoir un modèle simple et performant à la fois.

La suite de la section faisant appel à ces notions, il est important de les avoir à l'esprit.

Le modèle sous-jacent à l'algorithme XGBoost est l'agrégation d'arbres (ou boosting d'arbres). L'agrégation d'arbres est un modèle composé d'arbres de régression ou de classification.

Exemple d'agrégation d'arbres

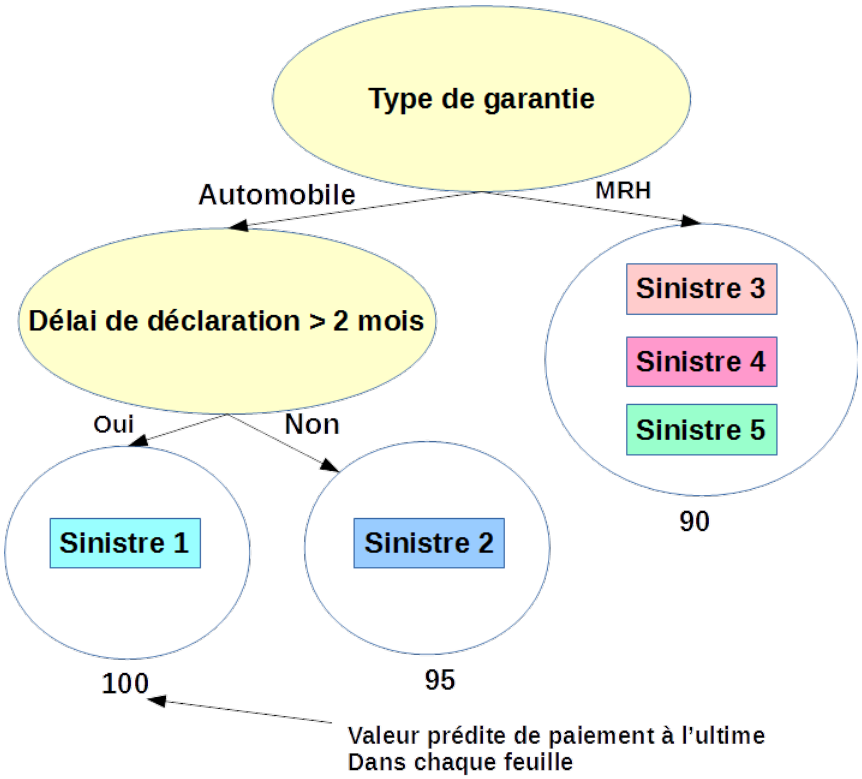


FIGURE 3.1 – Exemple d'arbre de régression

Dans le graphique ci-avant, des sinistres ont été classés et on leur a attribué un résultat, un score associé à la feuille à laquelle ils ont été affectés. La régression est effectuée à partir de deux variables illustratives : *Type de garantie* et *Délai de déclaration*. La valeur prédite correspond à la valeur de paiement à l'ultime estimée pour chaque sinistre.

Toutefois, le pouvoir prédictif d'un seul arbre n'est souvent pas suffisant pour résoudre un problème relativement complexe. C'est pour cela que l'on va utiliser un modèle d'agrégation d'arbres qui va sommer la prédiction de plusieurs arbres ensemble.

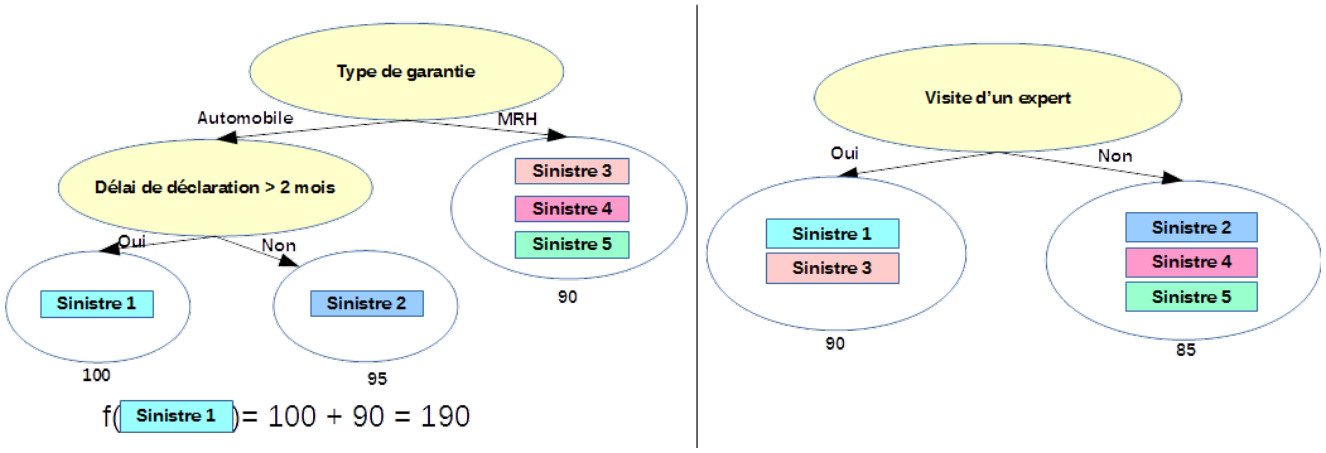


FIGURE 3.2 – Exemple de modèle d'agrégation d'arbres à 2 arbres

L'exemple précédent représente un modèle d'agrégation d'arbres constitué de 2 arbres. Le score associé à un sinistre pour chaque arbre est additionné pour obtenir le score final du sinistre. Ce qui est intéressant de noter est que chaque arbre complète le précédent.

Un peu de formalisme

Soit un ensemble de données $\mathcal{D}$ avec n individus et p variables explicatives : $\mathcal{D} = (x_i, y_i)_{1 \leq i \leq n}$ et $|\mathcal{D}| = n, x_i \in \mathbb{R}^p, y_i \in \mathbb{R}$. Un modèle d'agrégation d'arbres est un modèle additif qui utilise M fonctions (ici des arbres de régression) pour prédire le résultat final. On a donc :

$$\hat{y}_i = \sum_{k=1}^{M} f_k(x_i), f_k \in \mathcal{F} \tag{3.3}$$

où $\mathcal{F} = f(x) = w_{q(x)}$ $(w \in \mathcal{R}^T)$ est l'espace des arbres de régression.

De plus q $(q : \mathbb{R}^p \to T)$ représente la structure de chaque arbre qui rattache un exemple (i.e. une observation) à la feuille qui lui correspond. Le terme « structure » désigne la manière dont l'arbre sépare la population à chaque noeud. En outre, T est le nombre de feuilles de l'arbre.

Enfin, chaque fonction f_k correspond à un arbre indépendant de structure q avec T feuilles et w_i représente le score de la $i^{ème}$ feuille d'un arbre.

Pour rappel de la section précédente, la fonction de perte dans le cadre de l'algorithme XGBoost est l'erreur quadratique moyenne (notée l ci-après). Ainsi, avec un modèle additif à M fonctions, une prédiction définie comme en (3.3) et en reprenant l'écriture d'une fonction objectif présentée en (3.1), on obtient la fonction objectif suivante à optimiser dans le cadre d'une agrégation d'arbres :

$$obj(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{M} K(f_k) \tag{3.4}$$

Avec :

$$K(f_k) = \gamma T + \frac{1}{2}\lambda ||w||^2$$

où γ et λ sont des paramètres de pénalisation du modèle et $||\cdot||$ est la norme euclidienne. Cette définition du terme de régularisation est proposée par les auteurs de la méthode XGBoost et sera admise pour la suite de l'étude.

Le Gradient Boosting

Comme le suggère l'équation (3.4), la fonction objectif d'un modèle d'agrégation d'arbres utilise des fonctions comme paramètres, elle ne peut donc pas être optimisée avec les méthodes traditionnelles d'optimisation dans des espaces euclidiens. La stratégie consiste alors à raisonner de manière additive. À chaque itération, il faut apprendre de l'erreur commise à l'itération précédente et ajouter un arbre qui va réduire l'erreur de prédiction. C'est ainsi que fonctionne le boosting. Ainsi, la prédiction à l'étape t pour un individu i peut s'écrire de la manière suivante :

$$\begin{aligned}
&\hat{y}_i^{(0)} = 0 \\
&\hat{y}_i^{(1)} = f_1(x_i) = f_1(x_i) + \hat{y}_i^{(0)} \\
&\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = f_2(x_i) + \hat{y}_i^{(1)} \\
&\quad ... \\
&\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned} \tag{3.5}$$

De l'équation (3.5) on en déduit la fonction objectif à optimiser à l'étape t :

$$obj(\theta)^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + K(f_t) + \ constante \tag{3.6}$$

On cherche à chaque itération la fonction f_t qui permettra d'optimiser la fonction objectif définie en (3.6). Si l'on écrit le développement de Tayor à l'ordre 2 de la fonction définie en (3.6), on obtient :

$$obj(\theta)^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2] + K(f_t) + \ constante \tag{3.7}$$

où $g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ et $h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ sont les dérivés partielles respectivement d'ordre 1 et 2 de la fonction de perte l. L'équation (3.7) met en avant le fait que la fonction objectif à optimiser à l'itération t ne dépend que du gradient (d'où l'appellation Gradient boosting) et de la hessienne. C'est grâce à ce fait que l'algorithme XGBoost peut fonctionner avec différentes fonctions de perte (deux fois différentiables au minimum).

Enfin, en remplaçant K par sa valeur définie en (3.4) et en retirant les constantes de l'équation (3.7), on obtient la fonction objectif simplifiée suivante au temps t :

$$\begin{aligned}
obj(\theta)^{(t)} &= \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\
&= \sum_{j=1}^{T} [(\sum_{i \in S_j} g_i) w_j + \frac{1}{2} (\sum_{i \in S_j} h_i + \lambda) w_j^2] + \gamma T
\end{aligned} \tag{3.8}$$

où $S_j = \{i|q(x_i) = j\}$ est l'ensemble des individus affectés à la même feuille j pour un arbre de structure q. De là, pour une structure q donnée, et puisque les w_j sont indépendants entre eux, on peut déterminer le score optimal w_j^* pour la feuille j :

$$w_j^* = \frac{\sum_{i \in S_j} g_i}{\sum_{i \in S_j} h_i + \lambda} \tag{3.9}$$

On en déduit la valeur de fonction objectif optimale à l'étape t :

$$obj^{*(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in S_j} g_i)^2}{\sum_{i \in S_j} h_i + \lambda} + \gamma T \tag{3.10}$$

C'est cette dernière équation qui mesure la qualité de la structure, en d'autres termes, la qualité de l'arbre ajouté. Ainsi, plus cette valeur sera petite et plus l'ajout de l'arbre associé sera pertinent.

Maintenant que l'on possède un moyen de mesurer la qualité d'un arbre, il reste à déterminer quels arbres choisir. Il n'est pas possible dans la pratique de tous les lister et de tous les tester. Donc, l'algorithme cherche à optimiser l'arbre à chaque étape. Ainsi, si S_R et S_L sont les sous-groupes créés après une subdivision, alors on peut définir le gain de la division par :

$$Gain = \frac{1}{2}[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_R + G_L)^2}{H_R + H_L + \lambda}] - \gamma$$

où $G_R = \sum_{i \in S_R} g_i$ et $H_R = \sum_{i \in S_R} h_i$.

Ce gain peut être décomposé en 4 parties dans cet ordre :
- le score de la nouvelle feuille à gauche
- le score de la nouvelle feuille à droite
- le score du noeud original
- un terme de régularisation

Ce que l'on peut constater est que si le gain est plus faible que γ le terme de régularisation, alors il n'y a pas d'intérêt à effectuer cette subdivision. C'est ainsi que s'achève la présentation de l'algorithme XGBoost.

3.2.3 Application du modèle

Projection des charges à l'ultime

Choix de la fonction objectif

Rappel sur la famille exponentielle

Avant de présenter les éléments intervenant dans le choix de la fonction objectif, il est intéressant d'effectuer un rappel sur ce qu'est une loi appartenant à la famille exponentielle. Ainsi, toutes les distributions de la famille exponentielle peuvent être écrites avec la forme :

$$f(y|\theta, \phi) = a(y, \phi) exp(\frac{y\theta - k(\theta)}{\phi})$$

où :
- $a(\cdot)$ et $k(\cdot)$ des fonctions données

— ϕ est appelé paramètre de dispersion. C'est un paramètre de nuisance strictement positif (supposé constant et connu dans un premier temps). Quand une pondération w des individus existe, ce paramètre est remplacé par : $\frac{\phi}{w}$.
— θ est un paramètre réel appelé paramètre canonique ou encore paramètre naturel

Pour toute variable Y suivant une distribution appartenant à la famille exponentielle, il est possible d'écrire les moments de la façon suivante :

$$E[Y] = \mu = k'(\theta)$$
$$V(Y) = \phi k''(\theta)$$

En provisionnement, ou lorsque l'on souhaite décrire la distribution de la charge de sinistres en assurance, c'est souvent la distribution Gamma qui est utilisée. Cette fois ce ne sera pas le cas car il existe a priori une modélisation plus adaptée.

Distribution de Tweedie

Dans le domaine des probabilités et des statistiques, la distribution de Tweedie fait partie de la famille exponentielle tout comme les distributions normale et Gamma, la distribution de Poisson ou encore les distributions composées Poisson-Gamma qui ont une quantité importante de zéros. Soit Y une variable aléatoire suivant une distribution de Tweedie, alors il existe la relation suivante :

$$V(Y) = aE[Y]^p$$

Les variables a et p de l'égalité précédente sont des constantes positives. De la formule ci-avant on a que :
— $p = 0$ correspond à une distribution Normale
— $p = 1$ correspond à une distribution de Poisson
— $p = 2$ correspond à une distribution Gamma

Pour le bien de cette étude, on considère que p est compris entre 1 et 2. Dans ce cas on parle d'une distribution de Poisson composée avec des sauts Gamma, appelée par la suite distribution de Tweedie. Dans la pratique cette distribution possède des particularités qui sont en accord avec notre problématique :
— Elle est positive. Les quelques cas marginaux ayant une charge totale négative ne seront pas utilisés pour la modélisation
— Une forte masse en zéro. Or, les garanties que nous souhaitons modéliser ont un pic important en zéro dû au très grand nombre de sinistres sans suite présents dans la base
— Elle est très asymétrique étalée à droite comme on a pu l'observer sur les données des charges

Comme évoqué précédemment, la distribution de Tweedie fait partie de la famille des lois exponentielles. On peut alors l'écrire sous la forme suivante :

$$f(y|\theta, \phi) = a(y, \phi)exp(\frac{y\theta - k(\theta)}{\phi}) \tag{3.11}$$

Avec $\mu = E[Y] = k'(\theta)$ et $V(Y) = \phi k''(\theta)$ d'après les propriétés évoquées précédemment sur la famille exponentielle et leurs moments. Si l'on pose $V(Y) = \phi\mu^p$, où $p \in]1, 2[$, alors on montre (en annexe A.1 et A.2) que :

$$k''(\theta) = \mu^p$$

$$\theta = \frac{1}{1-p}\mu^{1-p}$$

$$k(\theta) = \frac{1}{2-p}\mu^{2-p}$$

On peut alors réécrire l'équation (3.11) sous la forme suivante :

$$f(y|\theta,\phi) = a(y,\phi)exp(\frac{1}{\phi}(\frac{y\mu^{1-p}}{1-p} - \frac{\mu^{2-p}}{2-p}))$$

Le paramètre ϕ est constant dans le cas de notre étude. De plus, on supposera aussi que $\log(\mu) = f(x)$ (avec $f(x)$ la prédiction faite pour l'individu x avec l'arbre de régression f).

Enfin la log-vraisemblance de Tweedie peut être exprimée de la façon suivante :

$$\ell = log(f(y|\theta,\phi)) = \sum_i^n log(a(y_i,\phi)) + \frac{1}{\phi}(\frac{y_i\mu_i^{1-p}}{1-p} - \frac{\mu_i^{2-p}}{2-p}) = \sum_i^n log(a(y_i,\phi)) + \frac{1}{\phi}(\frac{y_i \exp^{(1-p)f(x_i)}}{1-p} - \frac{\exp^{(2-p)f(x_i)}}{2-p}) \tag{3.12}$$

Le détail de cette opération est disponible en annexe A.3.

Choix des paramètres

Comme évoqué précédemment, l'optimisation de l'algorithme XGBoost est long et complexe du fait d'un très grand nombre de paramètres. Le choix initial d'optimisation de l'algorithme s'est donc porté sur trois paramètres dont l'influence forte sur les modèles d'arbres sans agrégation pourrait également se répéter dans le cas d'une méthode agrégée. Ces paramètres sont :

- η qui est un paramètre d'élagage. Il permet de limiter la taille des arbres. Il est équivalent au coefficient γ défini plus haut.
- *subsample* est un paramètre qui détermine le pourcentage d'individus de l'échantillon à utiliser à chaque itération pour construire un classifieur
- *colsample_bytree* est un paramètre qui détermine le pourcentage de variables à utiliser à chaque itération pour construire un classifieur

On appelle les paramètres précédents des hyperparamètres. En effet, ils représentent des propriétés globales de l'algorithme qui ne sont pas entraînées pendant la phase d'apprentissage. Ces paramètres sont en effet déterminés avant de commencer le processus de calibrage de l'algorithme.

L'optimisation de ces paramètres a pour but d'éviter un phénomène de surapprentissage [1]. Et, afin d'éviter le surapprentissage, un échantillon de validation est fourni à l'algorithme. Le modèle apprend alors sur l'échantillon d'apprentissage et teste ensuite ce modèle sur l'échantillon de validation. À chaque itération, la valeur de l'erreur commise sur l'échantillon de validation et sur l'échantillon d'apprentissage est affichée. Par construction, plus le nombre d'arbres ajoutés est important et plus l'erreur sur l'échantillon d'apprentissage est faible. Cependant, si dans le même temps l'erreur sur l'échantillon test ne diminue pas aussi, alors le modèle est en surapprentissage. Dans la pratique, l'échantillon de validation choisi est l'échantillon test

1. Un modèle est dit en surapprentissage lorsqu'il apprend les détails et le bruit associé à un échantillon d'apprentissage. La conséquence est alors une piètre qualité de la prédiction sur un nouveau jeu de données.

Optimisation bayésienne

Afin d'obtenir les paramètres optimaux, nous allons calibrer le modèle grâce à une recherche par optimisation bayésienne. Ce calibrage permettra de choisir les valeurs optimales pour les paramètres évoqués précédemment.

L'optimisation bayésienne fonctionne en construisant une distribution a posteriori qui représente le mieux la fonction que l'on souhaite optimiser. Dans le cas de cette approche, l'objectif est de maximiser la fonction cible. L'approche teste différentes observations dans l'espace des variables impliquées dans l'optimisation. Plus le nombre d'observations est important, plus la distribution a posteriori est proche de la vraie fonction objectif. L'algorithme devient alors de plus en plus sûr des régions dans l'espace des paramètres qui valent la peine d'être explorées pour trouver l'optimum. Le schéma suivant traduit la description précédente à l'aide d'un exemple.

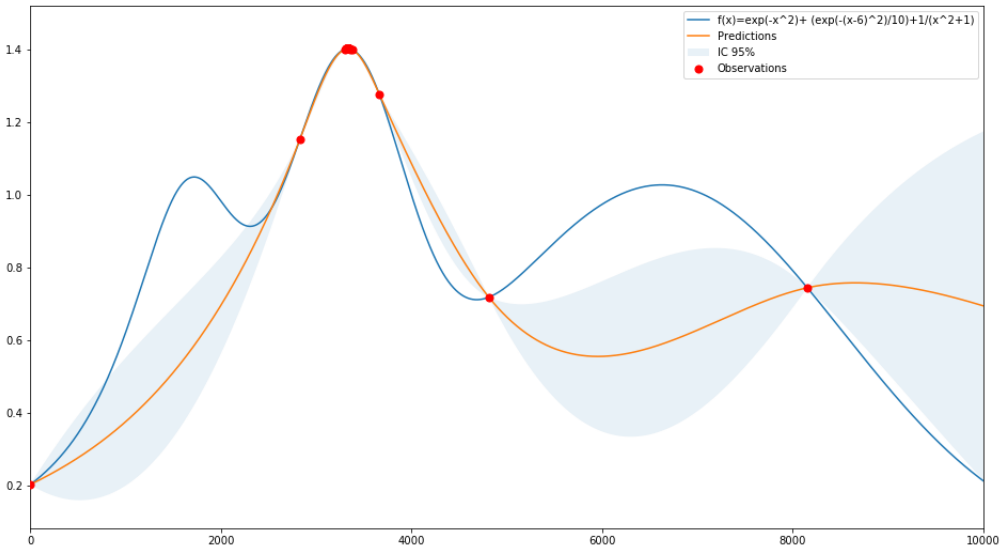


FIGURE 3.3 – Exemple d'optimisation bayésienne

La fonction objectif à estimer la suivante : $f(x) = e^{-x^2} + \frac{-(x-6)^2}{10} + \frac{1}{x^2+1}$.

Elle est décrite par la courbe en bleu. Les points en rouge correspondent aux points déjà explorés. Comme on peut le voir, les intervalles de confiance autour des zones peu explorées sont très grands. L'algorithme peut donc estimer qu'il serait intéressant de chercher des observations dans cette zone (concept d'exploration). Ou, connaissant déjà certains points, l'algorithme décide de se concentrer sur les zones connues (concept d'exploitation).

Ainsi, à chaque nouvelle itération, l'algorithme arbitre entre exploration et exploitation, en prenant en compte l'information dont il dispose sur la vraie fonction objectif. Un processus gaussien est calibré sur les points explorés précédemment. Ensuite, la distribution a posteriori, combinée à la stratégie d'exploration et d'exploitation est utilisée pour déterminer quel doit être le nouveau point à estimer.

Pour plus d'information sur cette approche, il est possible de se référer aux références de la bibliographie ([4], [5]).

Dans le cas de notre étude, la fonction à optimiser a été l'erreur quadratique moyenne. Les valeurs optimales ont été testées afin de vérifier leur performance avec la métrique utilisée par l'algorithme. Toutefois, la distribution de Tweedie est fortement marquée par des valeurs proches de 0, elle a donc un peu plus de difficultés lorsque les montants sont très élevés. Un arbitrage entre un modèle très optimal théoriquement et un peu moins en pratique est donc nécessaire.

Finalement, les paramètres choisis sont :
— $\eta = 0.1$
— *subsample*= 1
— *colsample_bytree* = 1
Les autres paramètres sont laissés à leur valeur par défaut.

Application

La métrique utilisée par l'algorithme XGBoost pour mesurer la qualité d'un modèle est la log-vraisemblance négative de Tweedie. Plus précisément, on remplace la fonction objectif générale définie en (3.6) par :

$$obj(\theta)^{(t)} = \underset{f_t \in \mathcal{F}}{\text{argmin}}\{-\ell\} = \underset{f_t \in \mathcal{F}}{\text{argmin}} \sum_{i=1}^{n} -l(y_i, f_t(x_i)), \tag{3.13}$$

avec la log-vraisemblance l de Tweedie définie en (3.12) et pour laquelle on suppose ϕ connu, $\mathcal{F}$ l'ensemble des arbres de régression et

$$l(y_i, f_t(x_i)) = \frac{y_i \text{exp}^{(1-p)f_t(x_i)}}{1-p} - \frac{\text{exp}^{(2-p)f_t(x_i)}}{2-p}$$

Ainsi, à chaque itération, on cherche la fonction qui va minimiser moins la log-vraisemblance définie en (3.13).

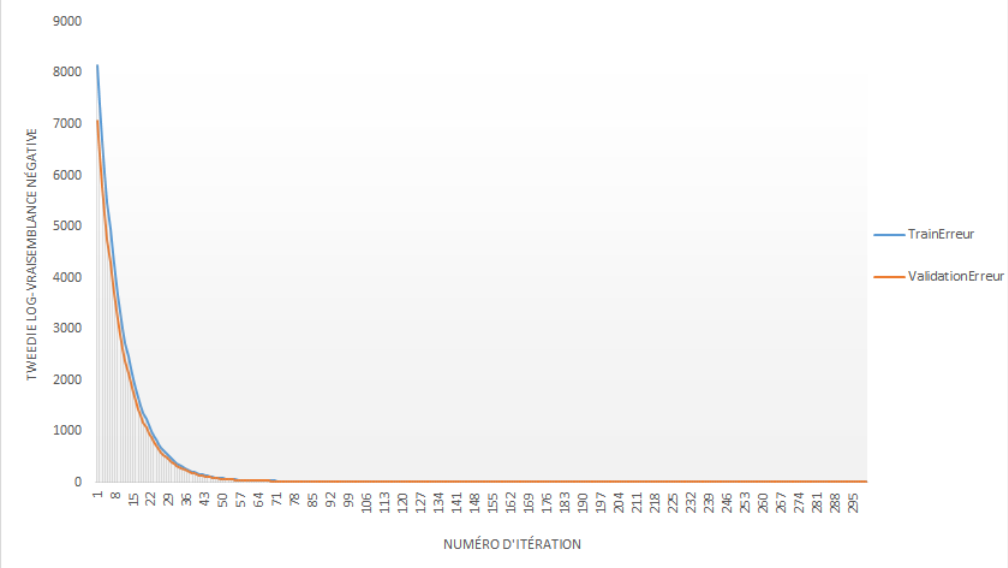


FIGURE 3.4 – Évolution de la log-vraisemblance négative de Tweedie, pour p=1.9

Le graphique précédent présente l'évolution complète de cette log-vraisemblance négative au fur et à mesure des itérations. On observe que la métrique converge vers des valeurs proches de 0 quand on se rapproche de 300 itérations. De plus, le gain le plus important est effectué au cours des 50 premières itérations. Enfin, la courbe orange, qui est celle de l'évolution de la métrique pour l'échantillon de validation, est toujours au-dessus de la courbe bleue tout en restant très proche. Il semble donc que le phénomène se surapprentissage ait été limité.

```
[1]     val-tweedie-nloglik@1.9:7062.716309     train-tweedie-nloglik@1.9:8151.895020
[101]   val-tweedie-nloglik@1.9:25.608124       train-tweedie-nloglik@1.9:25.713175
[201]   val-tweedie-nloglik@1.9:24.752037       train-tweedie-nloglik@1.9:24.609333
[300]   val-tweedie-nloglik@1.9:24.611338       train-tweedie-nloglik@1.9:24.435564
```

FIGURE 3.5 – Extrait de console R après exécution de l'algorithme XGBoost

La figure précédente est une capture d'écran de l'exécution de l'algorithme XGBoost pour 300 arbres (i.e. 300 itérations) et pour $p = 1.9$. Une nouvelle fois, on observe que l'erreur commise diminue à chaque itération pour l'échantillon d'apprentissage et pour l'échantillon de validation. De plus, on constate que la métrique utilisée (la log vraisemblance négative) est très petite : 25 environ pour les deux échantillons. En effet, il est coutumier de chercher les paramètres qui maximisent la log-vraisemblance d'un modèle. Il est donc équivalent de chercher ceux qui minimisent son opposé. Ce rappel effectué, il semble donc que le modèle ait une bonne qualité de prédiction puisque la valeur de la log-vraisemblance négative est petite.

Enfin, on constate que la valeur de la métrique est sensiblement la même pour l'échantillon de validation et pour l'échantillon d'apprentissage. La performance de l'algorithme est toutefois meilleure sur l'échantillon d'apprentissage. Cela n'est pas surprenant car l'échantillon d'apprentissage sert à calibrer le modèle. On pouvait donc s'attendre à ce que l'algorithme soit plus performant sur la base de calibration.

Analyse des résultats

Rappels sur la base

Afin de bien comprendre les résultats présentés par la suite, il est important d'avoir à l'esprit la manière dont est structurée la base.

Invariant | Variable | A prédire

Trimestre de développement	Numero de sinistre	Date sinistre	Date déclaration	Lieu	Cause sinistre	Nature dommage	Gravité dommage	Montant paiement	Autres variables	Etat du sinistre	Charge totale à l'ultime
0	333	01/01/14	01/01/14	63	Accident voiture	entorse genou	1	50,00	...	Ouvert	3000
3	333	01/01/14	01/01/14	63	Accident voiture	entorse genou	2	150,00	...	Ouvert	3000
6	333	01/01/14	01/01/14	63	Accident voiture	ligaments croises	7	200,00	...	Ouvert	3000
9	333	01/01/14	01/01/14	63	Accident voiture	ligaments croises	7	2 600,00	...	Ouvert	3000
12	333	01/01/14	01/01/14	64	Accident voiture	ligaments croises	7	0,00	...	Ouvert	3000
15	333	01/01/14	01/01/14	65	Accident voiture	ligaments croises	7	0,00	...	Ouvert	3000
18	333	01/01/14	01/01/14	66	Accident voiture	ligaments croises	7	0,00	...	Ouvert	3000
21	333	01/01/14	01/01/14	67	Accident voiture	ligaments croises	7	0,00	...	Ouvert	3000
24	333	01/01/14	01/01/14	68	Accident voiture	ligaments croises	7	0,00	...	Ouvert	3000
27	333	01/01/14	01/01/14	69	Accident voiture	ligaments croises	7	0,00	...	Ouvert	3000
30	333	01/01/14	01/01/14	70	Accident voiture	ligaments croises	7	0,00	...	Ouvert	3000
33	333	01/01/14	01/01/14	71	Accident voiture	ligaments croises	7	0,00	...	Ouvert	3000
36	333	01/01/14	01/01/14	72	Accident voiture	ligaments croises	7	0,00	...	Ouvert	3000
39	333	01/01/14	01/01/14	73	Accident voiture	ligaments croises	7	0,00	...	Ouvert	3000
42	333	01/01/14	01/01/14	74	Accident voiture	ligaments croises	7	0,00	...	Ouvert	3000
45	333	01/01/14	01/01/14	75	Accident voiture	ligaments croises	7	0,00	...	Ouvert	3000
48	333	01/01/14	01/01/14	76	Accident voiture	ligaments croises	7	0,00	...	Ouvert	3000
51	333	01/01/14	01/01/14	77	Accident voiture	ligaments croises	7	0,00	...	Fermé	3000
54	333	01/01/14	01/01/14	78	Accident voiture	ligaments croises	7	0,00	...	Fermé	3000
57	333	01/01/14	01/01/14	79	Accident voiture	ligaments croises	7	0,00	...	Fermé	3000
0	111	01/07/14	03/07/14	2	Accident voiture	entorse cervicale	3	100,00	...	ouvert	1500
3	111	01/07/14	03/07/14	3	Accident voiture	entorse cervicale	3	250,00	...	ouvert	1500
6	111	01/07/14	03/07/14	4	Accident voiture	entorse cervicale	3	0,00	...	ouvert	1500
9	111	01/07/14	03/07/14	5	Accident voiture	entorse cervicale	3	0,00	...	ouvert	1500
12	111	01/07/14	03/07/14	6	Accident voiture	entorse cervicale	3	0,00	...	ouvert	1500
15	111	01/07/14	03/07/14	7	Accident voiture	entorse cervicale	3	0,00	...	ouvert	1500
18	111	01/07/14	03/07/14	8	Accident voiture	entorse cervicale	3	0,00	...	ouvert	1500
21	111	01/07/14	03/07/14	9	Accident voiture	entorse cervicale	3	150,00	...	ouvert	1500
24	111	01/07/14	03/07/14	10	Accident voiture	entorse cervicale	3	0,00	...	ouvert	1500
27	111	01/07/14	03/07/14	11	Accident voiture	entorse cervicale	3	0,00	...	ouvert	1500
30	111	01/07/14	03/07/14	12	Accident voiture	entorse cervicale	3	0,00	...	ouvert	1500
33	111	01/07/14	03/07/14	13	Accident voiture	entorse cervicale	3	0,00	...	ouvert	1500
36	111	01/07/14	03/07/14	14	Accident voiture	coup du lapin	8	1 000,00	...	ouvert	1500
39	111	01/07/14	03/07/14	15	Accident voiture	coup du lapin	8	0,00	...	ouvert	1500
42	111	01/07/14	03/07/14	16	Accident voiture	coup du lapin	8	0,00	...	ouvert	1500
45	111	01/07/14	03/07/14	17	Accident voiture	coup du lapin	8	0,00	...	ouvert	1500
48	111	01/07/14	03/07/14	18	Accident voiture	coup du lapin	8	0,00	...	ouvert	1500
51	111	01/07/14	03/07/14	19	Accident voiture	coup du lapin	8	0,00	...	ouvert	1500
54	111	01/07/14	03/07/14	20	Accident voiture	coup du lapin	8	0,00	...	ouvert	1500
57	111	01/07/14	03/07/14	21	Accident voiture	coup du lapin	8	0,00	...	ouvert	1500

FIGURE 3.6 – Schéma de la structure de la base de données

Ci-dessus se trouve un schéma représentatif de la base de données (que ce soit l'échantillon d'apprentissage ou l'échantillon test). On trouve dans cette table deux sinistres hypothétiques représentés par leur numéro (dans l'exemple 333 et 111). Et chaque sinistre possède 20 lignes correspondant à ses 20 premiers développements dans le temps.

Certaines variables (en bleu clair sur le schéma) sont invariantes quelque soit le développement. Il s'agit par exemple des données telles que la date de survenance du sinistre ou la date de déclaration.

D'autres variables en revanche varient dans le temps (en bleu foncé sur le schéma), il peut s'agir du montant des paiements effectués par l'assureur sur la période ou encore la gravité du dommage. Ce sont ces variables qui font différer les lignes d'un même sinistre au cours du temps. Si l'on se ramène à notre problématique, ce sont ces lignes qui doivent refléter le plus fidèlement possible l'information dont dispose l'assureur concernant un sinistre à un instant donné. Il peut bien entendu s'agir d'un paiement comme évoqué. Mais les échanges avec l'assuré, l'intervention d'experts quel qu'ils soient constituent également des informations à disposition de l'assureur.

La variable à prédire (en rouge sur le schéma) est la charge totale à l'ultime. Elle est la même pour chaque ligne d'un même sinistre. C'est aussi à ce niveau que se situe l'originalité de l'approche.

L'idée est que l'algorithme devrait être capable de prédire la charge totale ultime quelque soit la période de développement où l'on se situe. Évidemment, cette valeur devrait évoluer lorsque les variables évoluent au fur et à mesure du temps. Ce serait un outil tout à fait pratique de pouvoir, à chaque clôture trimestrielle, estimer de manière précise la charge totale à payer à terme et d'en déduire les provisions en retranchant les paiements déjà effectués.

Comparaison aux données réelles

		Estimation par XGBoost									
	Valeur Réelle	Trimestre de développement									
Montants en euros		0	3	6	9	12	15	18	21	24	27
Charge Totale	16 053 652	8 424 575	10 115 620	11 576 860	12 783 817	13 525 178	14 304 705	14 385 070	15 217 256	15 432 906	15 707 834
Erreur relative	0%	-48%	-37%	-28%	-20%	-16%	-11%	-10%	-5%	-4%	-2%

		Estimation par XGBoost									
	Valeur Réelle	Trimestre de développement									
Montants en euros		30	33	36	39	42	45	48	51	54	57
Charge Totale	16 053 652	15 977 251	15 813 516	15 398 760	15 682 881	15 418 093	15 663 790	15 501 444	15 171 191	15 580 824	15 688 781
Erreur relative	0%	0%	-1%	-4%	-2%	-4%	-2%	-3%	-5%	-3%	-2%

FIGURE 3.7 – Analyse de l'erreur globale d'estimation avec XGBoost

L'erreur relative dans le tableau ci-dessus est affichée par apport aux montants réels. Ainsi, une valeur négative signifie que l'algorithme sous-estime et inversement.

Comme rappelé précédemment, l'algorithme est capable de prédire la charge totale ultime d'un sinistre à chaque étape de son développement. Les développements sont exprimés en trimestre à partir de la date de survenance. Ainsi le pas de temps 0 correspond au trimestre d'occurrence du sinistre. Le pas de temps 3 correspond lui au trimestre suivant le trimestre d'occurrence du sinistre et ainsi de suite.

La table précédente présente la charge totale de sinistralité vue à aujourd'hui en fonction du pas de développement considéré au moment de la prédiction. Ainsi le montant de 8 424 575 euros pour le trimestre de développement 0 correspond au montant de la charge totale ultime estimée, vue à aujourd'hui, en n'utilisant que le premier pas de développement (trimestre 0) pour chaque sinistre. Pour rappel, compte tenu de l'hypothèse d'horizon de 5 ans pour l'ultime, cette charge totale estimée est celle des sinistres dont la survenance est antérieure à 2012.

L'algorithme converge vers une solution proche de la réalité quand on utilise des pas de temps de plus en plus récents. En effet, de 48% d'écart (en valeur absolue) pour une projection utilisant le premier pas de temps, l'algorithme finit avec une erreur relative de 2% en utilisant le pas de temps le plus récent. Intuitivement, cette observation est plutôt logique car en utilisant le trimestre le plus récent, l'algorithme se rapproche de la vision à l'ultime supposée du sinistre. Il paraît donc cohérent que l'erreur soit moins importante.

En analysant plus en détails, on constate de grandes disparités dans les prédictions. Pour une prédiction effectuée dans les 18 premiers mois du sinistre, l'erreur relative globale est de plus de 10%. De manière plus précise, l'algorithme sous-estime de manière très forte la charge totale de sinistre ultime. Puis, une tendance haussière se développe et pour une prédiction faite après 18 mois de développement, l'algorithme a tendance à sous-évaluer la charge totale ultime. Toutefois, l'erreur globale commise reste par la suite toujours inférieure à 5%.

Afin de pouvoir comparer les estimations des différentes méthodes, la dernière diagonale du triangle des charges vue à fin 2012 a été projetée à l'aide de l'algorithme XGBoost. Grâce à ces projections, des valeurs de provisions ont été calculées en réalisant la différence entre la valeur estimée à l'ultime et la charge réelle observée à fin 2012.

Provision observée	Provision estimée avec XGBoost
-4 845 953 €	-5 011 681 €

montants en euros

FIGURE 3.8 – Comparaison des provisions estimées (avec XGBoost) et des provisions réelles sur l'échantillon test

En projetant la charge de sinistres vue à fin 2012, la provision globale estimée par XGBoost est proche de la provision réelle observée. En effet, l'erreur globale commise n'est que de 3% environ. Toutefois, on constate que l'algorithme a tendance à être trop optimiste plus qu'il a prévu une baisse de la charge totale plus importante que ce qui a pu être observé dans la réalité. C'est important car dans le cas d'une utilisation dans un cadre réglementaire, il faudrait sans doute rajouter une marge de risque supplémentaire au montant de provision estimée.

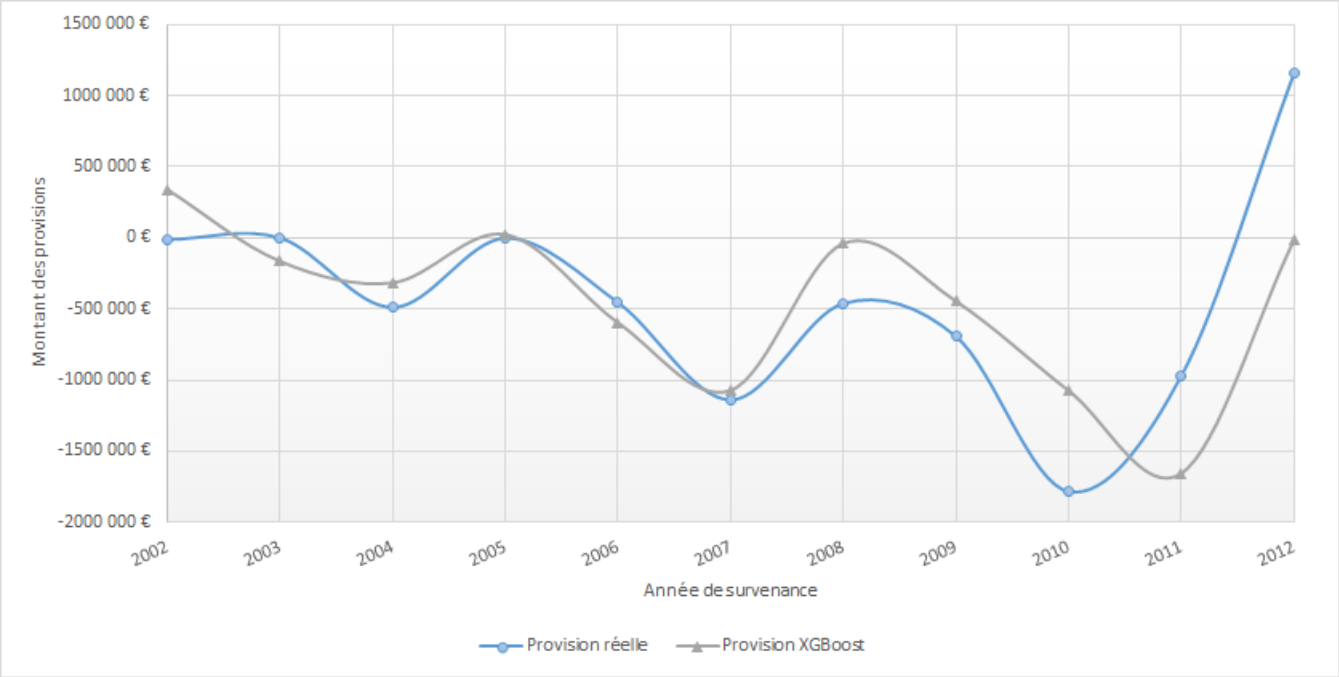


FIGURE 3.9 – Comparaison des charges totales par Chain-Ladder sur l'échantillon test

Le graphique ci-avant présente pour chaque année de survenance, le montant des provisions estimées par XGBoost (triangles gris) à la provision réelle (en bleu).

Ce que l'on peut constater est la relativement bonne performance de l'algorithme sur les sinistres ayant une date de survenance jusqu'en 2009. En effet, graphiquement on constate que les points gris et bleu sont confondus, ou en tout cas très proches.

Si globalement les estimations sont plutôt correctes, il existe toutefois quelques disparités importantes lorsque l'on analyse un peu plus en détails. En effet, les plus grandes erreurs de prédiction ont lieu sur les sinistres ayant les dates de survenance les plus récentes, à savoir les années 2010, 2011 et 2012. Il semble donc que l'algorithme ait du mal à prédire les sinistres les plus récents. De manière plus précise, le tableau suivant détaille les écarts entre les estimations de provision par la méthode XGBoost et la charge totale réelle.

Année de survenance	Provision observée	Provision estimée avec XGBoost
2002	-11 970 €	339 870 €
2003	0 €	-164 520 €
2004	-488 232 €	-314 675 €
2005	-1 652 €	24 773 €
2006	-454 292,00 €	-596 011 €
2007	-1 138 121 €	-1 071 289 €
2008	-461 045 €	-39 057 €
2009	-692 705 €	-445 495 €
2010	-1 782 887 €	-1 073 863 €
2011	-969 772 €	-1 655 465 €
2012	1 154 722 €	-15 950 €
Total	**-4 845 953 €**	**-5 011 681 €**

montants en euros

FIGURE 3.10 – Analyse de l'erreur d'estimation avec XGBoost

Lorsque l'on étudie les résultats plus finement, on constate que jusqu'en 2009 l'écart entre les estimations et la réalité est limité à 200 000 euros (en valeur absolue). Ce montant, certes important permet d'avoir un ordre de grandeur sur l'erreur commise par la prédiction. Cette observation est toutefois à nuancer à cause de deux contre-exemples.

Tout d'abord, en 2002, où l'algorithme a prédit une hausse de près de 340 000 euros de la charge quand en réalité celle-ci a stagné, et même légèrement diminué. Pour ces sinistres, qui sont les plus anciens de la base, il semble donc que l'approche par XGBoost ne soit pas performante. Cette observation peut être mise en parallèle avec l'ancienneté des sinistres considérés qui ont alors plus de 13 ans. De ce fait, des impacts difficiles à estimer peuvent en partie expliquer ces écarts importants. Et n'ayant pas d'éléments dans la base concernant ces effets (monétaires par exemple), il est normal que l'algorithme puisse avoir du mal à les modéliser.

La seconde exception est l'année 2008 pour laquelle l'algorithme a commis une erreur de plus de 400 000 euros (en valeur absolue). Certains phénomènes économiques de l'époque ont pu entrer en jeu (la crise financière mondiale par exemple), il est également possible que certaines modifications dans les produits proposés par l'entreprise aient eu lieu.

Concernant l'erreur importante commise sur les sinistres les plus récents, i.e., avec une année de survenance entre 2010 et 2012, il est possible que cette erreur soit dû à un problème de modélisation. En effet, l'hypothèse de base fut de considérer que 5 ans suffisaient pour avoir une vue correcte et stable du sinistre. Peut-être qu'en réalité, quelques années supplémentaires d'information, surtout dans le cas de la GAV, seraient nécessaire. Ainsi, pour ces sinistres pour lesquels on a 5, 6 ou 7 ans d'information, il serait peut-être nécessaire d'avoir quelques années de développement en plus.

Comparaison avec Chain-Ladder

Afin de déterminer la robustesse de l'algorithme par rapport à une approche traditionnelle, les projections des charges totales effectuées avec le deux modèles sont présentées graphiquement. Ces projections sont comparées aux valeurs réelles.

Provision observée	Provision estimée avec Chain-Ladder	Provision estimée avec XGBoost
-4 845 953 €	-749 482 €	-5 011 681 €

montants en euros

FIGURE 3.11 – Comparaison de l'estimation des charges totales avec XGBoost et Chain-Ladder

D'une manière globale, la prédiction faite par XGBoost est plus proche de la réalité que celle réalisée avec Chain-Ladder. En effet, la prédiction par Chain-Ladder est trop pessimiste et sous-évalue le montant de la baisse des provisions de près de 77% par rapport à la valeur réelle. L'erreur commise par XGBoost n'est, elle, que de 3% environ au global. Cette écart de provision se traduit par une charge totale estimée à l'ultime de 20,1 millions d'euros pour Chain-Ladder et de 15,8 millions pour l'approche par XGBoost. La valeur réelle à posteriori étant d'un peu plus de 16 millions d'euros, l'approche par XGBoost semble donc plus performante au global.

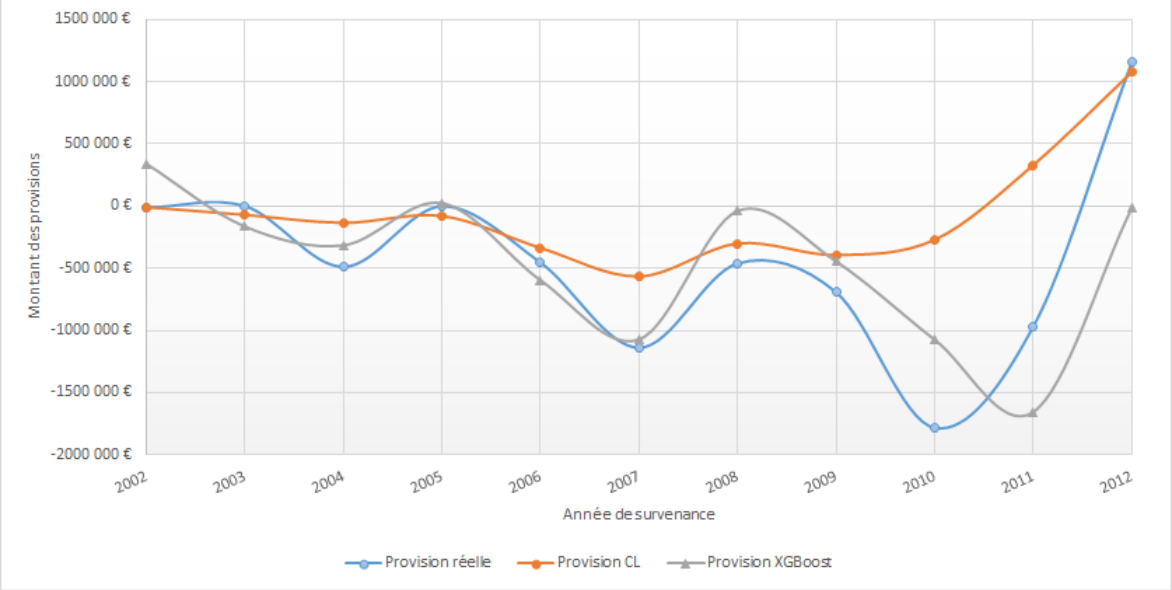


FIGURE 3.12 – Comparaison de l'estimation des charges totales avec Chain-Ladder et XGBoost

Le graphique précédent détaille les provisions vues à aujourd'hui en fonction des années de survenance de sinistres. Les ronds en bleu représentent les montants réels, les ronds en orange les estimations via Chain-Ladder et les triangles gris les prédictions de XGBoost.

La tendance globale de la charge réelle est à une légère baisse. Cette évolution globale semble bien être suivie par la méthode XGBoost alors que la technique de Chain-Ladder semble elle stagner au cours du temps.

Plus en détails, on a que les points orange sont majoritairement au dessus des ronds bleus. De là, on en déduit que la méthode de Chain-Ladder sous-estime presque systématiquement le montant de la baisse de la charge. Mais, au-delà d'être trop pessimiste, graphiquement, il semble que les estimations par Chain-Ladder soient le plus souvent les plus éloignées des valeurs réelles. Cette observation est vérifiée sauf pour les années de survenance 2002, 2008 et 2012. Il s'agit des périodes où l'algorithme XGBoost est le plus éloigné des valeurs réelles.

Le tableau ci-après compare année par année les prédictions des différentes techniques avec les données réelles.

Année de survenance	Provision observée	Provision estimée avec Chain-Ladder	Provision estimée avec XGBoost
2002	-11 970 €	-8 487 €	339 870 €
2003	0 €	-69 830 €	-164 520 €
2004	-488 232 €	-134 225 €	-314 675 €
2005	-1 652 €	-79 148 €	24 773 €
2006	-454 292,00 €	-336 112 €	-596 011 €
2007	-1 138 121 €	-564 787 €	-1 071 289 €
2008	-461 045 €	-300 691 €	-39 057 €
2009	-692 705 €	-392 928 €	-445 495 €
2010	-1 782 887 €	-270 554 €	-1 073 863 €
2011	-969 772 €	330 231 €	-1 655 465 €
2012	1 154 722 €	1 077 049 €	-15 950 €
Total	**-4 845 953 €**	**-749 482 €**	**-5 011 681 €**

montants en euros

FIGURE 3.13 – Comparaison entre les charges réelles et les charges estimées par Chain-Ladder et XGBoost

Le plus souvent, la méthode de Chain-Ladder surestime le montant de la provision à mettre en place. La technique de Chain-Ladder est donc trop pessimiste.

Enfin, malgré une plus grande volatilité, surtout pour les sinistres ayant une année de survenance ancienne, l'approche par XGBoost est tout de même plus fiable que Chain-Ladder. Dans plus de la moitié des cas prédits, l'estimation faite par l'algorithme XGBoost est plus fiable que celle de Chain-Ladder, i.e., l'erreur commise est plus faible (en valeur absolue).

Finalement, concernant la charge globale, l'algorithme XGBoost est bien plus performant que la méthode de Chain-Ladder. De la même manière, en analysant année par année, l'approche est légèrement plus performante que Chain-Ladder.

Analyse des variables significatives

Maintenant qu'il semble que le modèle ait montré sa fiabilité, il semble intéressant d'étudier les variables sur lesquelles l'algorithme s'est basé.

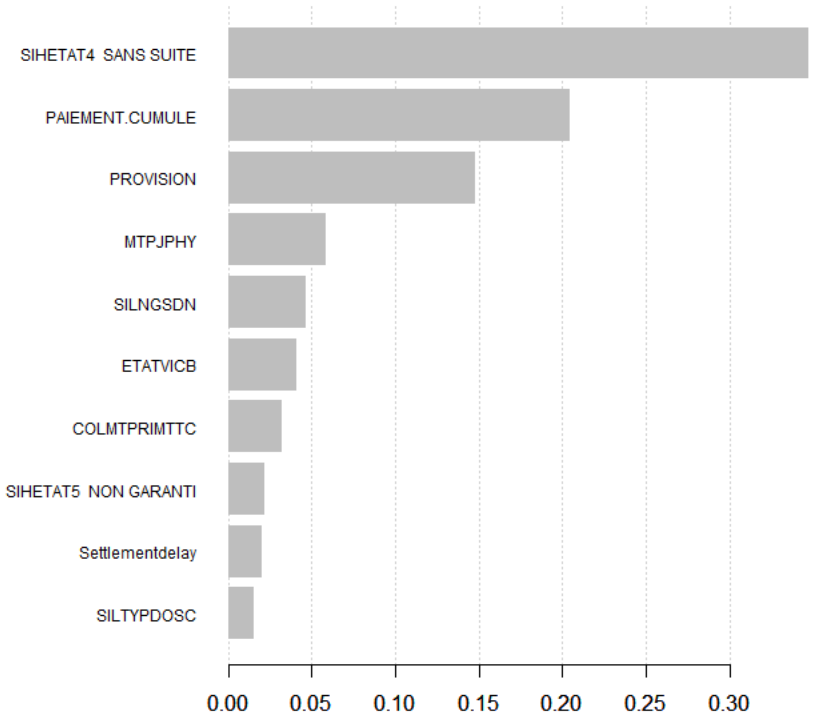


FIGURE 3.14 – Importance des 10 variables utilisés par la méthode XGBoost

Comme évoqué dans la description de la méthode, l'importance d'une variable correspond au gain en performance de l'algorithme lorsqu'il décide d'utiliser cette variable au moment de la construction des classifieurs (ici des arbres de décisions binaires). Ainsi plus une variable est utilisée et plus son gain global sera élevé.

De manière assez logique, parmi les trois variables majeurs pour l'algorithme, deux d'entre elles correspondent aux charges de l'assureur : les paiements cumulés et les provisions. Il semble tout à fait cohérent d'imaginer que si à une même période de développement donnée deux sinistres ont des montants de paiements cumulés significativement distincts alors leurs charges totales à l'ultime soient elles aussi distinctes. Un raisonnement similaire pourrait expliquer l'importance des provisions. Au-delà des différents forfaits de provision, à l'ouverture notamment, il paraît raisonnable de supposer qu'un sinistre avec une provision élevée ait une charge totale ultime élevée également. Toutefois, le montant des provisions étant parfois instable, une dépendance à cette variable doit donc être une source d'attention permanente.
Mais, cette étude ayant pour but de fiabiliser le calcul de ces provisions, il est raisonnable d'espérer que les montants renseignés pour cette variable soient cohérents.

L'état des sinistres joue également un rôle important. En effet on peut noter la présence des variables *sihetat4 sans suite* et *sihetat5 non garanti* dans le graphique précédent. Ces données indiquent si le sinistre est respectivement sans suite ou non garanti.

La pertinence de ces variables paraît logique au vue du jeu de données utilisée pour l'étude. En effet, une masse importante de charges à prédire étaient très proches de 0. Il semble que ces montants de charge totale correspondent en grande partie à des sinistres sans suite voire non garantis. Compte tenu de la distribution particulière de la variable cible et du lien étroit entre des montants et l'état du sinistre, il est donc normal que ces variables soient ressorties.

Dans le même ordre d'idée que les variables décrivant l'état du sinistre, l'algorithme a également donné une grande importance à la variable *settlementdelay*. Cette variable correspond à la durée de temps (lorsqu'elle existe) entre la date de déclaration du sinistre et la date à laquelle le sinistre est clôturé. Il semble donc que la durée de vie d'un sinistre soit un facteur important dans l'évaluation du montant de la charge totale à l'ultime.

La variable *colmtprimttc* correspond au montant de la prime toute taxe comprise associée au contrat auquel est rattaché le sinistre. Cette donnée est plutôt intéressante, car cela pourrait indiquer un besoin de revu des tarifs de la part de l'entreprise. Seule l'étude de l'historique des contrats ayant eu des sinistres pourraient raisonnablement éclairer les raisons de la présence de cette variable.

Si les données analysées ci-avant pouvaient avoir une résonance globale, i.e. quel que soit l'entreprise, celles restantes semblent elles être plus spécifiques au portefeuille de la compagnie. La variable *mtpjphy* correspond au montant associé à un déficit fonctionnel permanent (DFP) dans le cas d'un préjudice physiologique. La donnée *siltypdosc* est une variable qualitative qui correspond à une classification du dossier du sinistre. Enfin, l'état de la victime semble jouer un rôle puisque l'état B de la victime apparaît comme ayant un rôle déterminant dans le montant de la charge totale ultime.

Afin de mieux comprendre comment a fonctionné l'algorithme, voici le tout premier arbre qui a été construit.

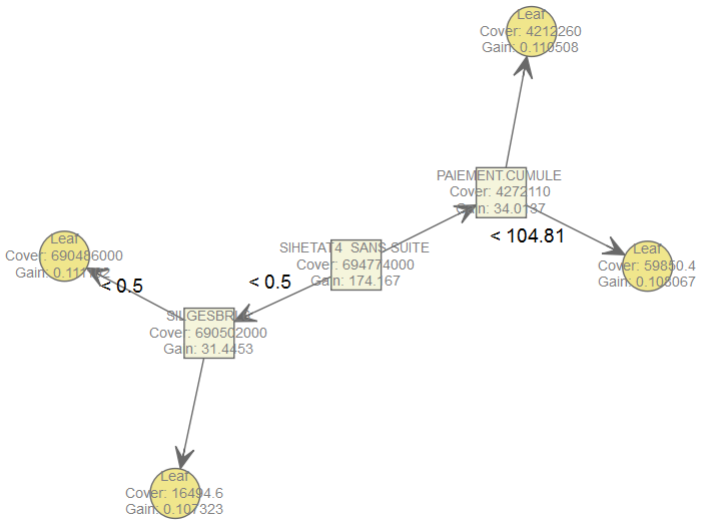


FIGURE 3.15 – Premier classifieur faible de l'algorithme XGBoost

L'algorithme a tout d'abord analysé l'état du sinistre. S'il était sans suite (dans ce cas la valeur de la variable était plus petite que 0,5) alors il passait un second test.

Ainsi, si la valeur du paiement cumulé était strictement inférieure à 104,81 euros alors le sinistre était classé dans une feuille. Sinon, il allait dans un autre feuille. Dans le même temps, si le sinistre était sans suite, (valeur supérieure à 0,5) alors l'algorithme testait si le gestionnaire dossier (variable *silges*) avait la valeur "BR" ou non. De la même façon que pour l'autre branche de l'arbre, si le gestionnaire dossier avait cette dernière valeur alors le sinistre était classé dans une certaine feuille. Sinon, il allait dans l'autre. Pour rappel, tous les sinistres d'une même feuille se voient attribuer le même score, dans notre cas la même valeur de charge totale ultime.

La valeur *gain* mesure l'importance du noeud dans le modèle. La valeur *cover* correspond à la somme des valeurs de la dérivée du second ordre des individus classés dans ce noeud (cf. présentation de la méthode XGBoost).

Conclusion sur l'utilisation de XGBoost

L'algorithme XGBoost est, de réputation, capable de s'adapter à de nombreuses problématiques. Il semble que ce soit confirmé dans le cadre de note étude. En effet, les résultats de la projection de la charge totale par XGBoost se sont montrés très positifs et encourageants. L'algorithme a montré qu'il ne commettait au global que peu d'erreur. Et ce, même sur un échantillon d'individus qui n'avait pas servi à sa calibration. De plus, l'utilisation d'arbres de décisions comme classifieurs a l'avantage opérationnel de permettre l'affichage des variables les plus importantes pour le modèle. Cet aspect permet de communiquer plus aisément sur les résultats de l'algorithme et vient en partie compenser le côté "boîte noire" de l'approche.

L'algorithme repose cependant sur une structure mathématique extrêmement complexe. De plus, son grand nombre de paramètres rend extrêmement difficile son optimisation. En effet, il serait trop coûteux en temps de calcul d'essayer d'optimiser tous les paramètres en même temps. Il faut dont avancer de manière progressive. Toutefois, dans la pratique, seuls les hyper-paramètres sont optimisés afin de rendre l'optimisation plus optimale.

Le temps de calcul peut également croître de manière très rapide si l'on décide d'utiliser des arbres ayant une grande profondeur. Ensuite, mais c'est une remarque valable pour la plupart de ces nouvelles approches automatisées, un pré-traitement des données est nécessaire. Celui-ci peut prendre une ampleur plus ou moins importante en fonction des contraintes propres à l'algorithme utilisé. Enfin, si l'on souhaite éviter la philosophie des Shadok [2] qui dit que « Ce n'est qu'en essayant continuellement que l'on finit par réussir.... En d'autres termes... Plus ça rate et plus on a de chances que ça marche... », alors il peut être utile d'avoir des connaissances de base en mathématiques et en informatique. Cela permet, d'essayer immédiatement les solutions sans doute les plus adaptées. Toute fois, une fois calibrée, le modèle est très simple à réutiliser.

Finalement, grâce à son adaptabilité à un grand nombre de problématiques, grâce la possibilité d'afficher les variables les plus significatives et grâce à la possibilité d'optimiser l'algorithme de manière assez fiable via les hyperparamètres, l'algorithme XGBoost semble être un outil très puissant et intéressant dans le cas du provisionnement.

2. Les Shadoks est une série télévisée d'animation française en 208 épisodes de deux à trois minutes, créée par Jacques Rouxel. Il s'agit de créatures construisant des machines improbables qui ne fonctionnent jamais.

3.3 Algorithme Long Short-Term Memory

Le Long short-term memory (appelé LSTM par la suite) repose sur les réseaux de neurones artificiels. Un réseau de neurones est un ensemble d'algorithmes dont le fonctionnement a été inspiré, schématiquement, par le fonctionnement des neurones biologiques. Par la suite, des méthodes statistiques sont venues transformer l'approche originale.

En se basant sur ces réseaux de neurones, Sepp Hochreiter et Jürgen Schmidhuber en 1997 (Sepp Hochreiter,Jürgen Schmidhuber [17]) et l'équipe de Felix Gers en 2000 (Felix A. Gers, Jürgen Schmidhuber, Fred Cummins [18]) ont pensé une architecture de neurones permettant de conserver au sein de l'algorithme la trace d'une information au cours de périodes de temps arbitrairement grandes. C'est ainsi qu'est apparue l'approche LSTM.

3.3.1 Lien avec notre problématique

Avantages de la méthode

L'un des avantages de la méthode LSTM est sa capacité à conserver une information au cours des différentes itérations de l'algorithme. Notre problématique est de déterminer, à partir d'un développement connu du sinistre, quel sera le montant de charge totale à l'ultime. L'hypothèse tacite est que le développement du sinistre au cours des cinq premières années de sa vie (s"il n'est pas clos) fournit suffisamment d'information sur la valeur de sa charge ultime. Ainsi, les nombreux changements qui peuvent affecter la vie d'un sinistre, comme le passage d'un expert ou l'évolution de la blessure de l'assuré devraient constituer des informations pertinentes quant à la prédiction de la charge ultime. Et par construction, l'algorithme saura conserver l'information que le niveau de blessure d'un assuré à changer et dans le même temps ne pas considérer certaines informations sans doute moins essentielles à la résolution du problème.

Un autre point fort de l'approche LSTM, plus opérationnel, est sa capacité à détecter des types de lien non linéaires entre les variables explicatives et la variable cible. Cet avantage, déjà évoqué pour l'approche XGBoost, est tout à fait important et explique la possibilité de l'approche de traiter des problématiques très diverses. Dans le même ordre d'idée, l'approche est susceptible de détecter les éventuelles interactions entre les variables explicatives elles-mêmes. Cet avantage que partagent de nombreuses approches de type machine learning permet de s'affranchir plus facilement du retraitement de certains liens entre les variables.

Du point de vue de l'implémentation, l'algorithme est capable de fonctionner en utilisant un nombre réduit d'hyperparamètres. Cela rend l'optimisation à priori un peu plus simple.

Limites de la méthode

Tout comme l'approche XGBoost, l'algorithme LSTM peut être considéré comme une boîte noire. Cette désignation est due en grande partie à la structure même de l'algorithme. En effet, par construction, l'algorithme LSTM repose sur une architecture de neurones dont il peut être, de prime abord, difficile de comprendre les interactions. Et, à cause des liens existants entre les différentes couches de la structure, il est de fait très difficile d'interpréter le rôle des variables dans le résultat final obtenu.

De plus, l'approche possède également les limites liées aux réseaux de neurones traditionnels. À savoir, ces techniques algorithmiques sont très sensibles à l'échantillon qui est fourni pour le calibrage. Ainsi, plus que d'autres approches, les réseaux de neurones requièrent un grand nombre de données pour être réellement efficaces. Il est important que l'algorithme ait vu le plus de situations différentes possibles si l'on souhaite que les résultats produits soient cohérents.

Outre un besoin important en quantité de données, celles-ci doivent également être de grande qualité. En effet, des données peu précises ou manquantes impliqueront des résultats erronés de la part de ces approches. Ainsi en 2015, à cause de données de mauvaises qualités, le service de reconnaissance d'images basé sur les réseaux de neurones d'un moteur de recherche connu avait malencontreusement classé des afro-américains en gorilles.

3.3.2 Les réseaux de neurones

Présentation d'un neurone

Afin de bien comprendre le fonctionnement de l'algorithme LSTM, il est important de revenir sur le fonctionnement d'un réseau de neurones.

Dans un soucis de cohérence, les notations de cette première partie seront valables tout le long de cette section, sauf mention contraire.

Un réseau de neurones est, en définitive, une très puissante fonction. On fournit à l'algorithme entraîné un vecteur en entrée et l'approche effectue une série d'opérations pour finalement produire un vecteur en sortie.

Plus précisément, un réseau de neurones est constitué par l'agrégation d'objets élémentaires, les neurones formels. Les réseaux se différencient en fonction de la structure du réseau : l'architecture (avec le nombre de couches, nombre de neurones par couche), le type de neurones (avec quelle fonction d'activation) et enfin leur objectif (apprentissage supervisé ou non, optimisation par exemple).

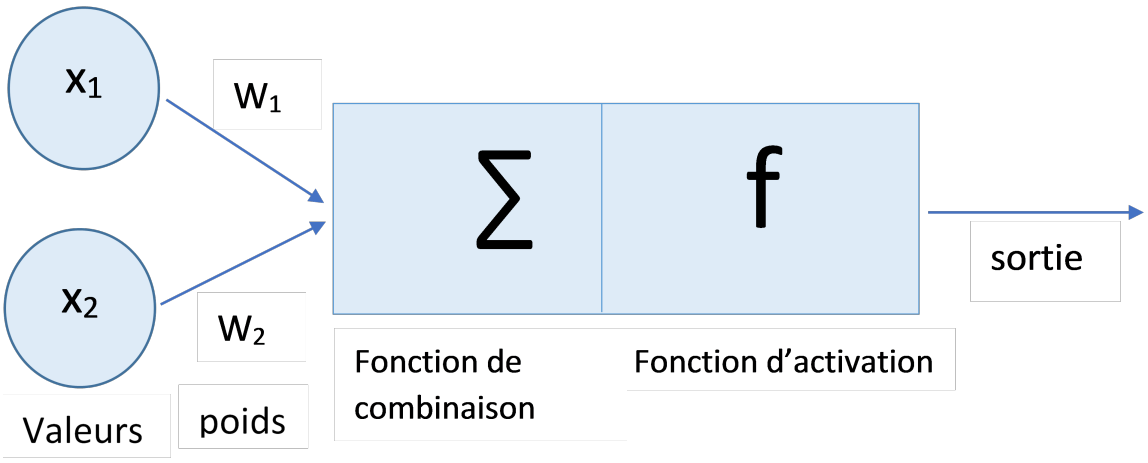


FIGURE 3.16 – Exemple de neurone formel

De manière très schématique, un neurone biologique est constitué par :
— **Des synapses** qui permettent de faire le lien entre les autres neurones, les fibres nerveuses ou musculaires
— **Des dentrites** qui jouent le rôle "d'entrées" du neurone
— **L'axone** qui est la sortie du neurone
— **Le noyau** qui active la sortie en fonction des simulations en entrée du neurone

C'est basé sur cette construction biologique que le neurone formel a été développé. Ainsi, comme on peut le voir sur le schéma précédent, à une série de valeurs (ou signaux) en entrée, on applique une **fonction d'activation**. Cette fonction effectue une transformation d'une combinaison des signaux fournis en entrée. Ainsi, le signal en sortie (y) d'un neurone formel peut être écrit de la façon suivante :

$$y = h(x_1, ..., x_n) = f(w_0 + \sum_{j=1}^{n} w_j x_j) = f(w_0 + w'x)$$

où f est la fonction d'activation, w est le vecteur des poids associés à chaque entrée et w_0 une constante représentant un biais introduit dans le modèle. Les valeurs des poids sont estimées pendant l'étape d'apprentissage. Ils représentent une forme de mémoire inhérente à l'algorithme.

Les types de neurones se différencient en fonction de leur fonction d'activation. Il en existe un grand nombre. Parmi les plus utilisés se trouvent les fonctions de type :
— **linéaire** où f est l'identité
— **sigmoïde** avec $f(x) = \frac{1}{1+e^x}$
— **tanh** avec $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

D'un neurone à un réseau de neurones artificiels

En utilisant un grand nombre de neurones formels, il est possible de créer ce que l'on appelle un *perceptron multicouche* (ou réseau de neurones "vanille"). Il s'agit d'un réseau de neurones composé de plusieurs couches successives. On peut définir une couche (ou "layer" en anglais) comme étant un groupe de neurones n'ayant pas de connexions entre eux.

Ainsi, une première couche reçoit les informations en entrée, avec un neurone par entrée. Une autre couche en sortie renvoie la réponse de l'algorithme. Enfin, une ou plusieurs couches cachées se trouvent entre la couche d'entrée et celle de sortie. Chaque neurone de ces couches intermédiaires est relié à tous les neurones de la couche précédente en entrée et à tous les neurones de la couche suivante en sortie.

Le schéma suivant représente de manière graphique la description précédente d'un perceptron multicouche.

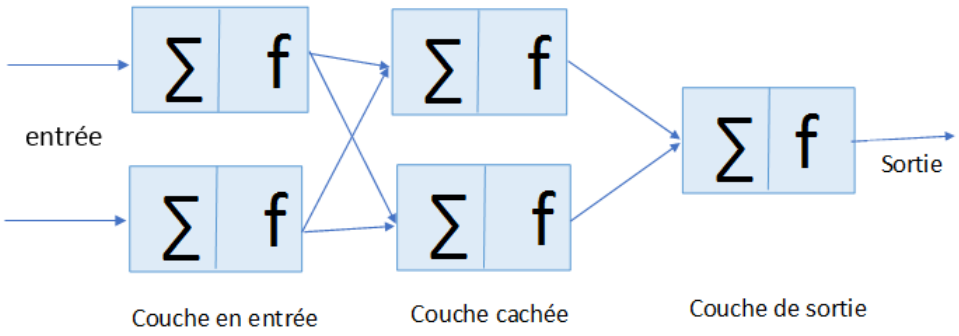


FIGURE 3.17 – Représentation d'un perceptron à une couche cachée de deux neurones

Enfin, de manière usuelle dans le cas d'une régression (i.e. la variable cible est quantitative comme dans notre cas), la dernière couche est constituée d'un seul neurone.

Apprentissage d'un réseau de neurones

L'apprentissage du réseau de neurones consiste à trouver les poids optimaux pour chaque couche du réseau de neurones afin de prendre la bonne décision en sortie.

Une nouvelle fois, pour pouvoir calibrer le modèle correctement, il est nécessaire d'avoir une métrique. Il s'agit donc de définir une fonction de coût. Et tout comme dans le cas de l'algorithme XGBoost, l'objectif sera de minimiser cette fonction objectif. Dans notre cas, la fonction objectif est la suivante :

$$obj(\theta) = \frac{1}{n}\sum_i (y_i - \hat{y}_i)^2$$

avec y la valeur réelle, $\hat{y}$ la valeur de la prédiction et n le nombre d'observations de la base de données. Il s'agit donc d'optimiser l'erreur quadratique moyenne.

L'algorithme d'optimisation utilisé le plus couramment est l'algorithme de descente de gradient. Cette approche n'assure que la convergence vers un minimum local. D'autres algorithmes d'optimisation sont souvent implémentés dans les programmes de réseau de neurones. Toutefois, l'objectif de ce mémoire n'étant pas d'étudier les algorithmes d'optimisation, c'est l'approche à l'aide de la descente de gradient que l'on utilisera.

Considérons un exemple simple avec un neurone (appelé neurone 3) qui prend deux entrées x_1 et x_2 et a une sortie s_3. La sortie s_3 sera considérée comme étant la prédiction. La situation décrite est représentée par le schéma suivant :

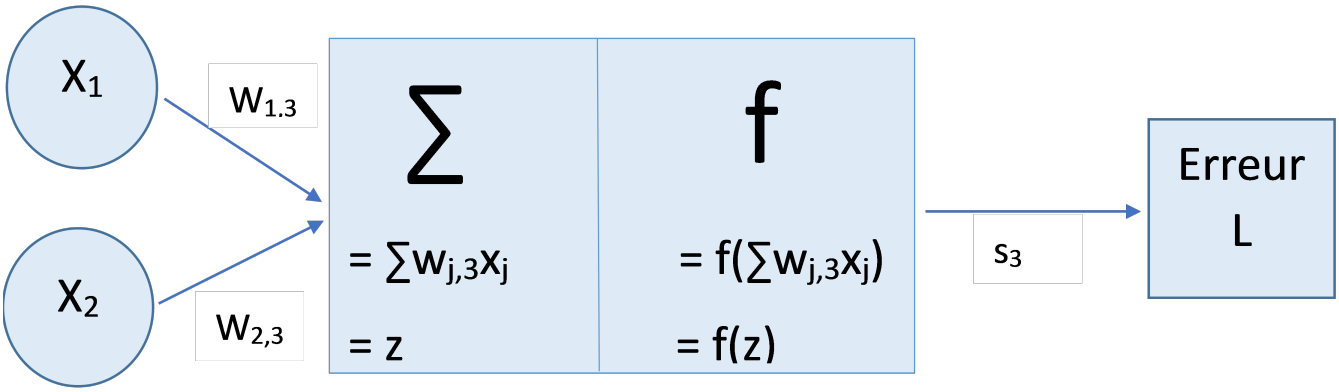


FIGURE 3.18 – Représentation du neurone 3

avec z la somme des entrées pondérées, $f(z)$ la transformation par une fonction d'activation de la combinaison linéaire z et L l'erreur calculée à partir de la prédiction s_3.

Ainsi, pour une valeur réelle à prédire y, on a cherche à minimiser la valeur suivante :

$$L = \frac{1}{2}(y - s_3)^2$$

L'utilisation de l'algorithme de descente nécessite le calcul de la dérivée partielle de l'erreur (notée L) en fonction du poids. Ainsi la dérivée partielle en fonction de $w_{1,3}$ vaut :

$$\frac{\partial L}{\partial w_{1,3}} = \frac{\partial L}{\partial s_3}\frac{\partial s_3}{\partial z}\frac{\partial z}{\partial w_{1,3}}$$

De plus, le premier facteur peut être réécrit de la manière suivante :

$$\begin{aligned}
\frac{\partial L}{\partial s_3} &= \frac{\partial}{\partial s_3}\frac{1}{2}(y - s_3)^2 && (3.14)\\
&= -\frac{1}{2}2(y - s_3) && (3.15)\\
&= -(y - s_3) && (3.16)
\end{aligned}$$

Le second facteur vaut :

$$\begin{aligned}
\frac{\partial s_3}{\partial z} &= \frac{\partial f(z)}{\partial z} && (3.17)\\
&= f(z)(1 - f(z)) && (3.18)\\
&= s_3(1 - s_3) && (3.19)
\end{aligned}$$

Enfin, le dernier facteur est équivalent à :

$$\begin{aligned}
\frac{\partial z}{\partial w_{1,3}} &= \frac{\partial(w_{1,3}x_1 + w_{2,3}x_2)}{\partial w_{1,3}} && (3.20)\\
&= x_1 && (3.21)
\end{aligned}$$

On a finalement que :

$$\frac{\partial L}{\partial w_{1,3}} = -(y - s_3)s_3(1 - s_3)x_1$$

Soit alors v_3 cette valeur qui ne dépend que du neurone numéro 3.

Considérons maintenant un perceptron multicouche à deux couches cachées de deux neurones chacune. Supposons que le neurone 3 précédent soit le neurone de sortie de ce perceptron. On a alors la structure suivante :

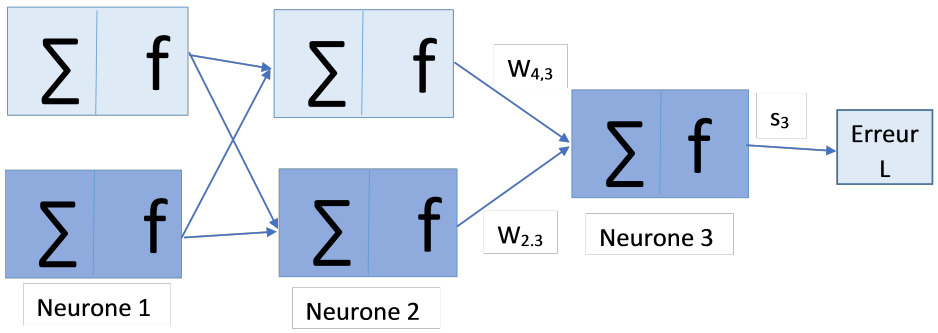
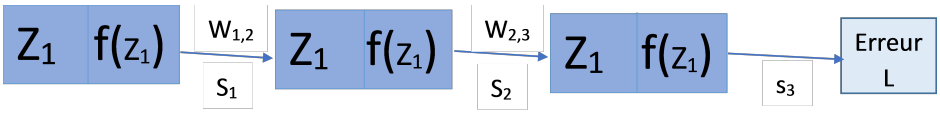


FIGURE 3.19 – Exemple de perceptron à deux couches cachées de deux neurones

Nous nous intéresserons particulièrement au chemin suivant :



La dérivée partielle de l'erreur L en fonction du poids $w_{1,2}$ est :

$$\frac{\partial L}{\partial w_{1,2}} = \frac{\partial L}{\partial s_2}\frac{\partial s_2}{\partial z_2}\frac{\partial z_2}{\partial w_{1,2}}$$

Et en utilisant le même raisonnement que précédemment on a finalement que :

$$\frac{\partial L}{\partial w_{1,2}} = v_3 \cdot w_{1,2} \cdot (y - s_2) \cdot s_3(1 - s_3) \cdot x_1$$

De la même façon, il est nécessaire de calculer les dérivées de la fonction objectif en fonction des poids de toutes les couches du réseau de neurones. On a alors toutes les dérivées partielles $\frac{\partial L}{\partial w_{i,j}}$ en fonction de tous les neurones i et j. On peut alors déterminer les poids optimaux en utilisant un algorithme de descente de gradient avec :

$$w_{i,j} = w_{i,j} - \alpha\frac{\partial L}{\partial w_{i,j}}$$

où α est le taux d'apprentissage (« learning rate » en anglais).

En résumé, le fonctionnement d'un réseau de neurones peut être défini en 2 étapes : une partie propagation "feedforward" et une partie "rétropropagation".

Ainsi, après une initialisation aléatoire des poids, une observation est fournie à la couche d'entrée. Puis à chaque itération, l'observation passe à travers tous les neurones du réseau couche après couche, de la couche d'entrée à une couche de sortie. Une prédiction est faite à la sortie de la couche de sortie. Il s'agit de la propagation "*feedforward*".

Dans en second temps, la prédiction est comparée à la valeur observée que l'on souhaite prédire. Puis, on remonte jusqu'à la couche d'entrée en passant par chaque couche en effectuant une mise à jour des poids. On parle alors de la "*rétropropagation*".

3.3.3 Les réseaux de neurones récurrents

Un peu de contexte

Un réseau de neurones artificiel (encore appelé réseau de neurones classique ou "vanille") est une structure algorithmique extrêmement puissante. En effet le théorème appelé "théorème d'approximation universelle" ([22]) prouve qu'une structure neuronale à une seule couche cachée est suffisante pour approcher avec une précision arbitraire toute fonction régulière dans un domaine fini de l'espace de ses variables. La démonstration de ce théorème n'ayant pas un aspect majeur de cette étude, elle ne sera pas abordée dans ce mémoire. Le lecteur pourra toutefois trouver des références dans la bibliographie.

Un réseau de neurones artificiels (noté ANN par la suite) peut donc reproduire à priori n'importe quel type de fonction régulière. Toutefois, il existe un certain nombres de tâches qu'un ANN ne peut pas résoudre. Ces problèmes incompatibles avec l'utilisation d'un ANN ont comme point commun l'usage de données séquentielles ou "temporelles" [3].

Afin de mettre en avant les lacunes d'un ANN, prenons l'exemple de la prédiction du sentiment associé à une phrase. Les valeurs en entrée de l'algorithme peuvent être considérées comme temporelles car il s'agit d'une séquence de mots. La sortie de l'algorithme est simple avec une valeur proche de 1 pour un sentiment positif et une valeur proche de 0 pour un sentiment négatif. Considérons la phrase suivante, plus certainement négative que positive :

« Some people just have bad luck » (*« Certaines personnes manquent simplement de chance »*)

Si l'on utilise l'approche ANN, alors il sera nécessaire de fournir à l'algorithme le mot "Some" en premier, puis "people", puis "just" et ainsi de suite. Mais à chaque propagation feedforward, la sortie serait inutilisable. En effet chaque sortie ne dépendrait que d'un seul mot uniquement. On chercherait alors le sentiment associé à un mot unique ce qui est peu pertinent. Dis autrement, cela revient à considérer que les observations fournies à chaque itération sont indépendantes, ce qui n'est pas réaliste dans le cas d'une phrase. Une phrase est seulement compréhensible si l'on met ensemble une série de mots dans un ordre spécifique pour lui donner un sens. La pertinence de chaque mot dépend alors des mots qui l'ont précédé : on parle du **contexte**. C'est pour cette raison que les réseaux de neurones récurrents (notés RNN par la suite) sont utilisés dans les problèmes à données séquentielles, ils peuvent retenir le contexte grâce à leur mémoire.

3. Une donnée temporelle est une information qui change au cours du temps

Toutefois, les RNN n'ont pas juste besoin de mémoire, ils ont besoin d'une mémoire à long terme (ou *long term memory*). Prenons cette fois l'exemple d'un mot à prédire connaissant le reste de la phrase :

Paul is late, he should ___. (« Paul est en retard, il devrait ___. »)

Si le réseau de neurones n'était pas en mesure d'avoir une mémoire à long terme (disons avant le mot « should »), différentes possibilités seraient alors envisageables :

Paul is late, he should fly. (« Paul est en retard, il devrait voler. »)
Paul is late, he should write. (« Paul est en retard, il devrait écrire. »)

Le mot « should » devrait donner l'information à l'algorithme qu'un verbe d'action est requis par la suite. Toutefois, si le RNN est capable de retenir l'information depuis son origine, i.e. depuis le début de la phrase, alors le mot « late » permettrait d'avoir une prédiction sans doute plus fiable :

Paul is late, he should hurry. (« Paul est en retard, il devrait se dépêcher. »)

La probabilité que l'algorithme puisse sortir le résultat « hurry » est d'autant plus forte lorsque le mot « late » a été observé au cours des itérations précédentes. C'est pour répondre à ce genre de problématiques que le contexte est très important. Les sorties d'un réseau de neurones récurrents seront toujours dépendantes de toute l'information fournie au travers des différentes itérations. Cependant, la manière dont l'algorithme retiendra ou oubliera une information dépendra de la manière dont les poids auront été calibrés durant la phase d'apprentissage.

Dans la pratique, les réseaux de neurones récurrents ont des difficultés à retenir une information sur de longues périodes. Cela est dû à un problème de disparition du gradient (« vanishing gradient problem » en anglais). Ce problème, développé dans le paragraphe suivant, justifie l'utilisation de variantes du RNN tel que l'algorithme LSTM.

Découverte non formelle d'un RNN

Pour rappel, voilà à quoi ressemble un réseau de neurones classique.

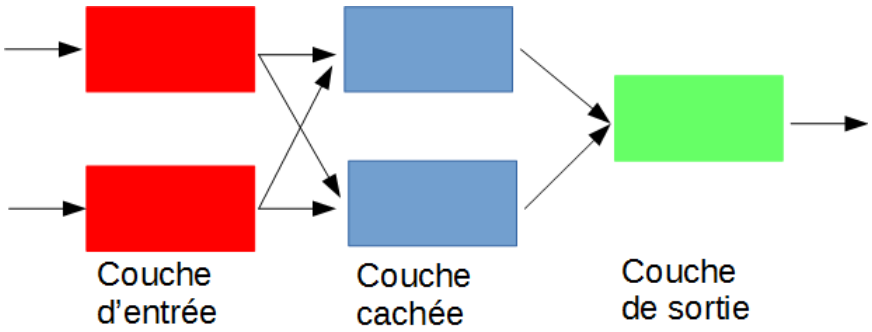


FIGURE 3.20 – Représentation d'un perceptron à une couche cachée de deux neurones

Chaque neurone ne stocke qu'une valeur unique. De ce point de vue, chaque couche peut donc être vue comme un vecteur.

Voici maintenant à quoi ressemble un RNN :

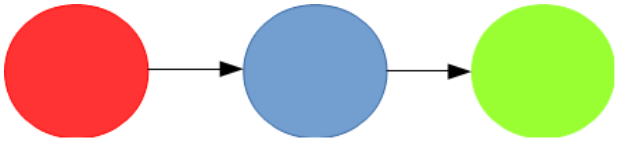


FIGURE 3.21 – Représentation réseau de neurones récurrents

Les deux schémas précédents représentent la même chose. Le second paraît toutefois un peu plus simple. Cela est dû à la visualisation du RNN dans laquelle chaque neurone (i.e. chaque rond) contient un vecteur d'information. Le terme de cellule est d'ailleurs parfaitement substituable au terme neurone dans le cas d'un RNN. Ainsi, en rouge se trouve le neurone d'entrée, en bleu un neurone caché et en vert un neurone de sortie. Finalement, une couche complète d'un ANN peut être représentée par un unique neurone dans le cas d'un RNN. Enfin, toutes les opérations dans le cas d'un RNN (comme la liaison d'un neurone à un autre) sont effectuées sur des vecteurs. Dans le cas d'un ANN, ces mêmes opérations étaient effectuées sur des scalaires.

Pour faciliter la compréhension du reste de la présentation, les RNNs seront présentés verticalement par la suite. On obtient alors :

FIGURE 3.22 – Représentation réseau de neurones récurrents

Le schéma précédent représente d'ailleurs un RNN « un pour un » (« one to one neuronal network » en anglais) car il lie un neurone d'entrée à un neurone de sortie. Il s'agit de l'architecture de réseau de neurones la plus simple.

Il est également possible d'avoir un réseau de neurones « un pour plusieurs » (« one to many » en anglais) où une entrée est liée à plusieurs sorties. Un exemple d'utilisation serait la reconnaissance

d'image où l'entrée serait une image (codée sous forme d'un vecteur) et la sortie serait une séquence de mots. Un tel RNN ressemblerait au schéma suivant :

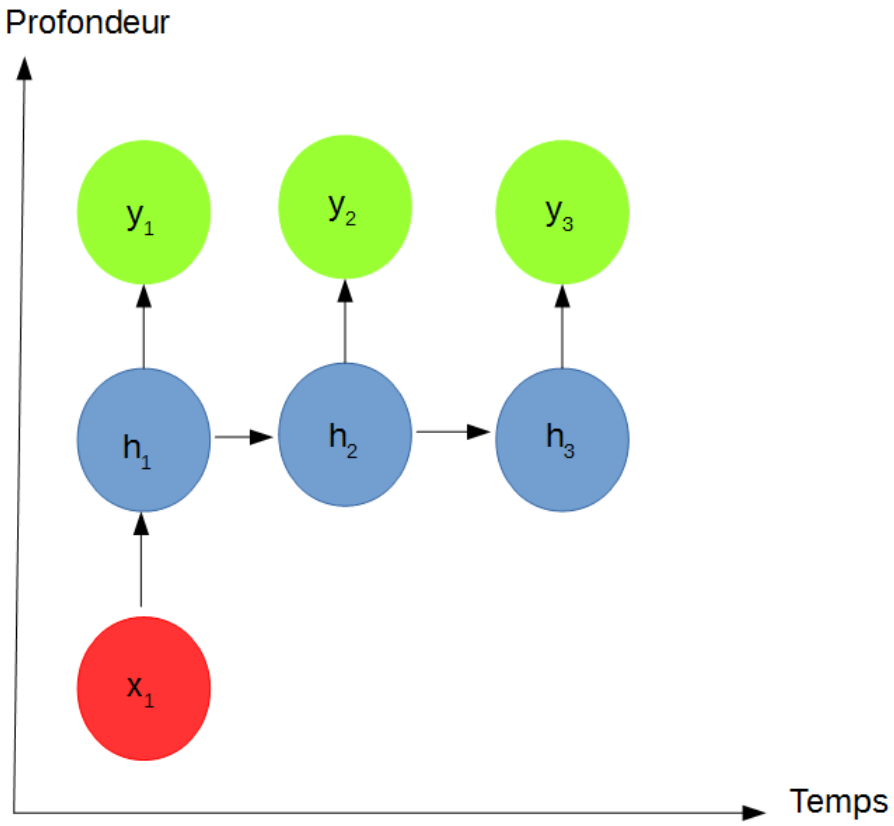


FIGURE 3.23 – Exemple de RNN « one to many »

Sur le schéma précédent, le temps en abscisse (découpé en pas de temps ou séquences) fait référence à l'ordre dans lequel les opérations sont effectuées par l'algorithme. La profondeur, en ordonnée, correspond aux liens entre la couche d'entrée, les couches cachées et la couche de sortie. Ainsi, plus le nombre de couches cachées augmente et plus la profondeur est grande. De la même façon, plus on progresse sur l'axe des abscisses et plus le nombre de pas de temps est important. Chaque point de ce graphique représente donc un neurone à un pas de temps donné et nourri par un neurone au temps précédent (que ce soit le même neurone ou un autre).

Contrairement à ce que laisserait penser le schéma, il n'y a pas 7 neurones dans ce réseau mais uniquement 3 : un neurone d'entrée, un neurone caché et un neurone de sortie. La différence avec un réseau de neurones classique est que chaque neurone expérimente maintenant une approche à « pas de temps multiples ». Et à chaque pas de temps, le neurone reçoit une nouvelle information sous la forme d'un vecteur.

Le déroulement du RNN dans le schéma précédent pourrait être décrit de la façon suivante :

1. Fournir x_1
2. Calculer h_1 en fonction de x_1
3. Calculer y_1 en fonction de h_1

4. Calculer h_2 en fonction de h_1
5. Calculer y_2 en fonction de h_2
6. Calculer h_3 en fonction de h_2
7. Calculer y_3 en fonction de h_3

Il serait également possible de calculer toutes les valeurs h_i des neurones cachées avant de calculer les sorties y_i (pour $i \in [1, 2, 3]$) associées.

Chaque sortie y est donc dépendante de toute l'information disponible depuis le début de la séquence. Dis autrement, l'algorithme RNN raisonne de la manière suivante :

> « Sachant ce que j'ai vu en entrée, sachant à quel pas de temps je suis, sachant encore ce que j'ai appris pendant l'apprentissage, je dois sortir le résultat y. »

Les sorties y sont donc dépendantes entre elles. Cependant, cette dépendance est indirecte. En effet, les sorties sont liées entre elles par les neurones cachées et pas directement entre elles (i.e. le RNN ne prédit pas y_2 directement à partir de y_1 mais partiellement à partir du neurone caché ayant prédit y_1).

Jusqu'à présent des réseaux de neurones récurrents « one to one » et « one to many » ont été présentés. Toutefois, dans notre cas, c'est un autre type de structure qui nous intéresse : les RNN « plusieurs pour un » (« many to one » neural network en anglais).

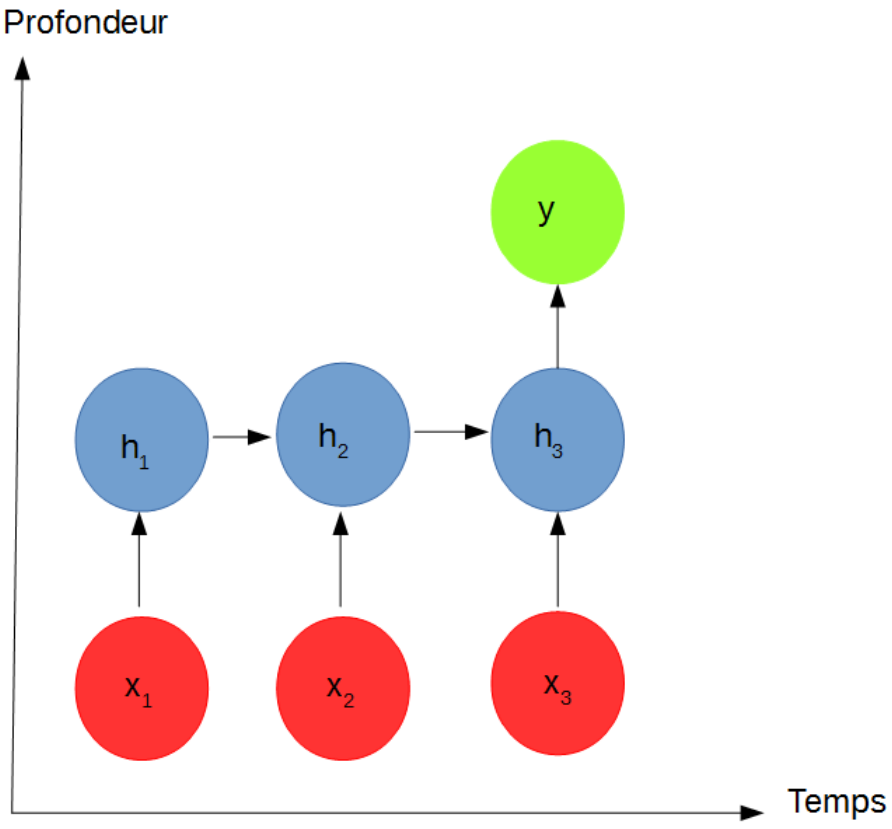


FIGURE 3.24 – Exemple de RNN « many to one »

Dans le cas d'un réseau « many to one », les entrées sont sous la forme d'une séquence. Ainsi, les neurones cachés sont à la fois dépendants de l'observation fournie à un pas de temps de donné et de l'information donnée par le neurone caché précédent. C'est pour cela que sur le graphique précédent, le second neurone caché (« h_2 ») a deux flèches en entrée.

C'est cette structure que nous utiliserons dans notre étude. En effet, les observations en entrée seront constituées par la vue du sinistre à différentes étapes de son développement. Le résultat attendu, unique, sera la valeur supposée de la charge à l'ultime du sinistre.

Enfin, la dernière structure possible est la « plusieurs pour plusieurs » (« many to many » en anglais) où les entrées et les sorties sont séquentielles.

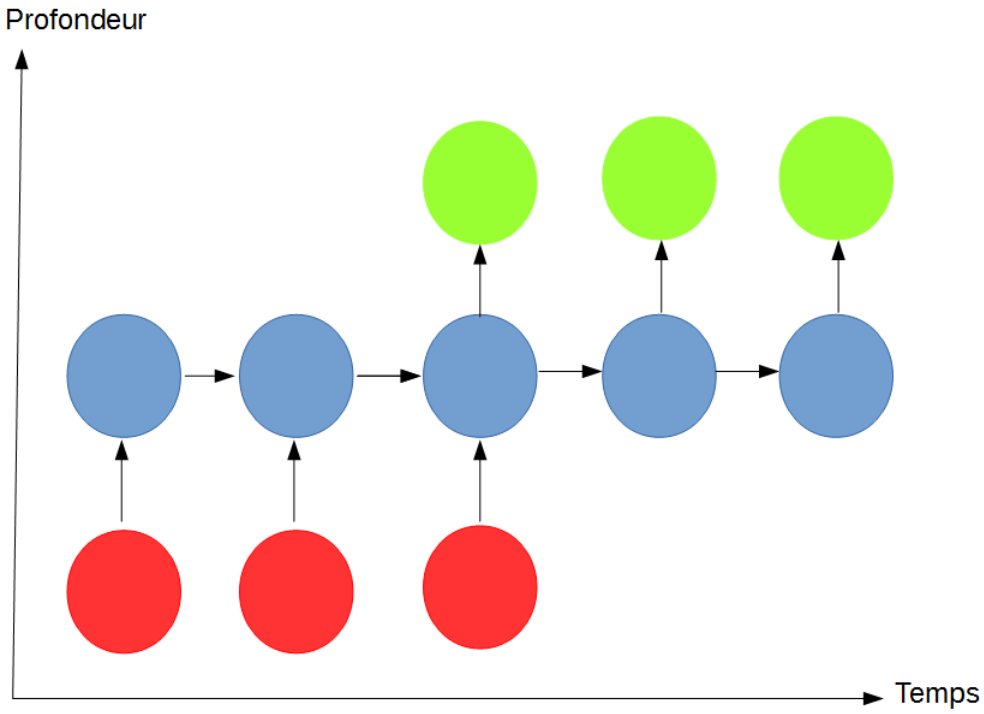


FIGURE 3.25 – Exemple de RNN « many to many »

Ce genre de structure pourrait tout à fait convenir à un problème de traduction où une séquence de mots en entrée a besoin d'être traduite en une autre séquence de mots. De la même façon, il est possible de construire une structure « many to many » avec plusieurs couches cachées. Chaque sortie d'un neurone pour une même couche est appelée un état caché (« hidden state » en anglais).

On peut encore présenter un RNN de la manière suivante :

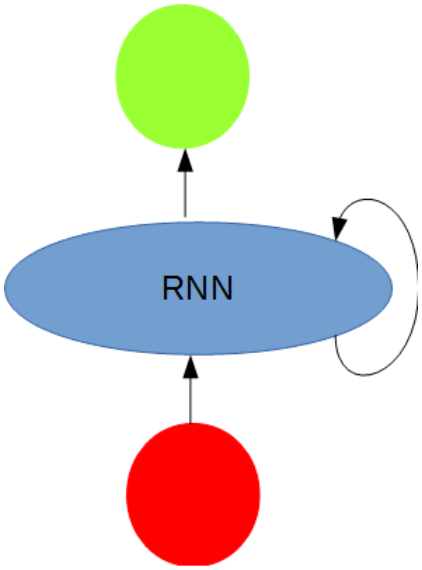


FIGURE 3.26 – Exemple de RNN « many to many »

Ce graphique est très significatif d'un RNN. En effet, on voit bien que l'algorithme prend une entrée, effectue une série d'opérations au cours du temps avant de retourner un résultat en sortie. La flèche à la fois en entrée et en sortie du RNN reflète bien l'idée d'une dépendance entre un neurone caché et le neurone caché du temps précédent.

Un pas dans la théorie

Dans cette section, les réseaux de neurones récurrents seront présentés de manière plus formelle.

Si un neurone (d'entrée ou de sortie) a une valeur au pas de temps t, on pose alors :

$x_t \leftarrow$ Entrée
$y_t \leftarrow$ Sortie

Concernant les neurones cachés, c'est un peu différent. En effet, puisqu'il peut exister plusieurs couches cachées, on définit le vecteur associé à une couche cachée au temps t pour la couche cachée l par :

$h_t^l \leftarrow$ Neurone cachée

Les entrées de ces neurones cachées sont des valeurs supposées connues. Les sorties sont elles inconnues. Afin que cette présentation soit générale, elle sera basée sur la structure neuronale la plus complexe.

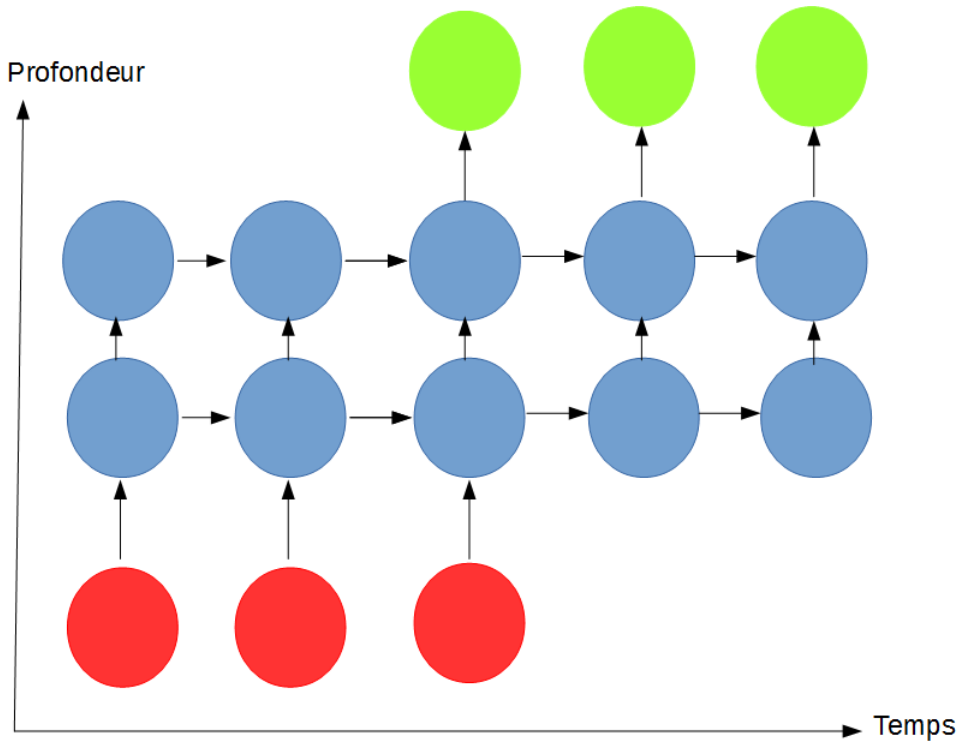


FIGURE 3.27 – Exemple de RNN « many to many »

Ainsi, ce RNN a une entrée séquentielle, une sortie séquentielle, plusieurs pas de temps et plusieurs couches cachées.

En regardant le schéma précédent, on constate qu'il peut exister plusieurs dépendances pour un neurone caché. On peut ainsi avoir :
— une entrée
— un état caché au temps précédent provenant de la même couche
— un état caché au temps actuel provenant d'une couche précédente

Un état caché peut avoir au maximum deux dépendances. Et en observant le graphique précédent, il est impossible d'obtenir une combinaison dépendant à la fois d'une entrée et d'un état caché au temps t actuel mais provenant d'une couche précédente. Dû à cette combinaison impossible, on peut définir deux équations distinctes : une équation pour un état caché sur la première couche cachée et une autre pour les couches cachées autres que la première. On a alors :

$$h_t^l = f_W(h_{t-1}^l, x_t), \text{ pour l=1} \tag{3.22}$$
$$= f_W(h_{t-1}^l, h_t^{l-1}), \text{ pour l>1} \tag{3.23}$$

avec f une fonction d'activation qui calcule l'état caché au temps t pour la couche l.

Le symbole W correspond aux poids du RNN. De manière plus précise, il existe cinq différentes matrices de poids qui correspondent aux différents types de liens dans la structure neuronale :

— entrée à neurone cachée $\rightarrow \mathrm{W}_{x,h}$
— neurone caché à neurone caché en fonction du temps $\rightarrow \mathrm{W}^{l}_{h,h,t}$
— neurone caché à neurone caché en fonction de la profondeur $\rightarrow \mathrm{W}^{l}_{h,h,d}$
— neurone caché à couche de sortie $\rightarrow \mathrm{W}_{h,y}$
— biais $\rightarrow \mathrm{b}^{l}_{h}$, b^{l}_{y}

Les indices de chaque matrice de poids indiquent quelle est leur utilité dans le réseau de neurones. Par exemple, le poids $W_{x,h}$ lie un vecteur d'entrée x à un un neurone caché h. Ou encore, la matrice $W^{l}_{h,h,d}$ lie un neurone caché h à un autre neurone caché h le long de l'axe vertical de la structure neuronale (i.e. la profondeur du réseau). En outre, comme les réseaux de neurones classiques, il est possible d'ajouter un biais représenté par le vecteur b_h ou b_y.

Pour les vecteurs b_h et même $W_{h,h,t}$, il y a différentes matrices de poids en fonction de la valeur de l. C'est parce que chaque couche cachée peut avoir un jeu de poids différents (cela serait en effet peu attractif si tous les poids de chaque couche étaient identiques), y compris le vecteur représentant le biais. Cependant, au sein d'un même neurone caché, tous les pas de temps partagent la même matrice de poids. C'est important car le nombre de pas de temps est une variable. Si par exemple on entraîne l'algorithme sur une séquence de 20 pas temps, mais qu'en pratique la séquence de sortie est constituée de 30 valeurs, il y aurait 10 valeurs en excès. Ainsi si chaque pas de temps avait une matrice de poids indépendante pour l'apprentissage, il ne serait pas alors possible de prédire ces 10 dernières valeurs car il n'y aurait pas de matrices de poids correspondantes. De la même façon, une matrice de poids pour chaque pas de temps impliquerait une croissance linéaire du nombre de paramètres en fonction de la taille de la séquence en entrée. Cela entraînerait très potentiellement du surapprentissage.

Définissons maintenant la fonction f_W :

$$f_W = f_W(h^{l}_{t-1}, h^{l-1}_{t}), \text{ pour } \mathrm{l} > 1 \tag{3.24}$$
$$= tanh(W^{l}_{h,h,t} h^{l}_{t-1} + W^{l}_{h,h,d} h^{l-1}_{t} + b^{l}_{h}) \tag{3.25}$$

$$f_W = f_W(h^{l}_{t-1}, x_t), \text{ pour } \mathrm{l} = 1 \tag{3.26}$$
$$= tanh(W^{l}_{h,h,t} h^{l}_{t-1} + W^{l}_{x,h} x_{tt} + b^{l}_{h}) \tag{3.27}$$

La fonction f_W est très similaire aux fonctions que l'on peut observer dans le cas d'un ANN : elle applique les bons poids aux paramètres correspondants, ajoute un biais et fournie la somme pondérée à une fonction d'activation. La différence est qu'il ne s'agit pas cette fois d'une somme pondérée mais d'un vecteur de sommes pondérées. Ainsi, chaque $W \times h$ (modulo le biais) aura la dimension d'un vecteur. La fonction tanh ressortira donc un vecteur où chaque valeur correspond à la valeur fournie en entrée transformée par la fonction tanh (avec l'ajout d'un potentiel terme de biais). La fonction tanh est très utilisée car la dérivée seconde ne s'annule pas rapidement, permettant de limiter le phénomène de disparition du gradient.

Enfin, l'équation liant un neurone caché à une sortie est :

$$y_t = W_{h,y} h_t^l + b_y \tag{3.28}$$

Il s'agit une équation possible, elle n'est toutefois pas unique. En fonction du contexte, il est possible de retirer le vecteur représentant le biais ou d'appliquer une transformation non linéaire telle que la fonction sigmoïde.

Le problème de disparition du gradient (ou « vanishing gradient problem »)

Tout comme pour un ANN, l'apprentissage d'un RNN consiste à trouver les poids optimaux aux extrémités de chaque couche. L'idée est qu'à chaque itération durant l'apprentissage, les poids soient perturbés (à l'aide d'une approche telle que la descente de gradient par exemple) de telle sorte que la probabilité que l'algorithme produise le bon résultat augmente à chaque fois.

Les sorties de l'approche neuronale sont comparées aux valeurs réelles à prédire à chaque pas de temps. Il est alors possible de calculer l'erreur commise par l'algorithme. Ici, on est dans le cas d'une propagation feedforward classique. Et, s'il existe n sorties différentes, chacune des sorties participe à l'erreur globale du modèle, il sera donc nécessaire de rétro-propager ces erreurs de manière individuelle et de les sommer ensuite. On a donc, grâce à la linéarité de la dérivation, pour un poids w quelconque et une fonction de perte L :

$$\frac{\partial L}{\partial w} = \sum_{k=1}^{n} \frac{\partial L}{\partial y_k} \frac{\partial y_k}{\partial w} = \frac{\partial L}{\partial y_1} \frac{\partial y_1}{\partial w} + \frac{\partial L}{\partial y_2} \frac{\partial y_2}{\partial w} + ... + \frac{\partial L}{\partial y_n} \frac{\partial y_n}{\partial w}$$

Cependant, dans le cadre d'un RNN, calculer ces gradients n'est pas une chose simple. Il existe un grand nombre de poids contribuant à la sortie, et il faut donc déterminer pour chacun sa contribution exacte et comment les modifier afin de réduire l'erreur globale. Pour répondre à cette problématique, il est d'usage de faire appel à la rétropropagation, comme pour un ANN. Dans le cas d'un RNN cependant, la rétropropagation est appelée « rétropropagation à travers le temps » (ou « backpropagation Through Time (BPTT) »). Il s'agit de la même approche qu'avec un ANN, sauf que cette fois c'est utilisé sur un RNN.

Il est important de garder à l'esprit que les RNN sont des modèles très complexes et profonds. En effet, en plus d'avoir plusieurs neurones cachés, chaque neurone caché a en plus de nombreux pas de temps. Ainsi si chaque couche caché d'un RNN a une centaine de pas de temps, cela reviendrait à construire un ANN avec des centaines de couches cachées, ce qui serait immense.

Afin de mieux saisir ce problème, prenons un exemple. Imaginons que l'on souhaite rétro-propager l'erreur jusqu'au temps 1 dans un RNN à k pas de temps. La dérivée serait de la forme suivante :

$$\frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_k} \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial h_{k-1}}{\partial h_{k-2}} \frac{\partial h_{k-2}}{\partial h_{k-3}} ... \frac{\partial h_2}{\partial h_1}$$

De plus, comme on a pour tout x réel : $h(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \in]-1, 1[$ donc $\frac{\partial h}{\partial x} = 1 - tanh^2(x) \in]0, 1[$.

Ensuite, pour obtenir la dérivée de l'erreur en fonction de la matrice des poids $W_{h,h}$, il sera nécessaire d'ajouter chacune des erreurs de gradient des états cachés par rapport au poids. De cette façon il sera alors possible de rétro-propager l'erreur en fonction des poids :

$$\frac{\partial L}{\partial W^l_{h,h}} = \frac{\partial L}{\partial h_k}\frac{\partial h_k}{\partial W^l_{h,h}} + \frac{\partial L}{\partial h_{k-1}}\frac{\partial h_{k-1}}{\partial W^l_{h,h}} + \frac{\partial L}{\partial h_{k-2}}\frac{\partial h_{k-2}}{\partial W^l_{h,h}} + ... + \frac{\partial L}{\partial h_1}\frac{\partial h_1}{\partial W^l_{h,h}}$$

Cela revient donc à ajouter un nombre important de termes qui tendent vers 0 avec le temps (car chaque terme est la composée du produit de facteurs plus petits que 1). Ainsi, le ratio $\frac{\partial L}{\partial W^l_{h,h}}$ ne capturerait en réalité que le signal des gradients provenant des pas de temps les plus récents. Et puisque la variation du gradient devient très proche de 0, il est alors nécessaire d'effectuer un très grand nombre d'itérations pour pouvoir converger à l'aide de l'algorithme de descente de gradient. C'est là que se situe le problème de disparition du gradient.

3.3.4 Le passage à l'approche LSTM

L'élément de base d'un RNN est le neurone caché. Il s'agit de l'unité qui prend des entrées puis à la fin d'un certain nombre d'itérations « temporelles », retourne des sorties. Plus précisément, à chaque pas de temps un neurone caché calcule un état caché en utilisant le vecteur de somme pondérée qui lui est fourni en entrée et/ou l'état caché précédent modifié d'un biais.

Avec les LSTM, il existe également des états cachés. Mais ceux-ci sont calculés à l'aide d'un mécanisme bien plus complexe : les cellules LSTM. Ainsi, au lieu de calculer chaque état caché comme une fonction directe des entrées et d'autres états cachés, les états sont calculés comme une fonction des valeurs issues des cellules LSTM (« cellule d'état ») au pas de temps considéré. Chaque cellule d'état est dépendante de la cellule d'état précédente et de toute autre information disponible.

Soit c_t la cellule d'état à un pas de temps t donné. Tout comme un état caché, une cellule d'état est juste un vecteur.

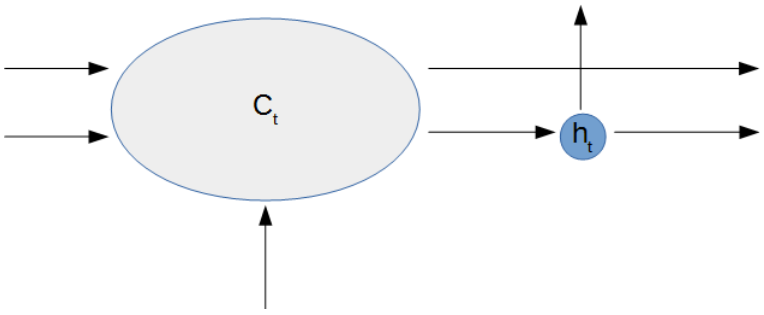


FIGURE 3.28 – Architecture LSTM au pas de temps t

Comme on peut le voir sur le schéma, une cellule LSTM reçoit de l'information de trois sources (représentées par les trois flèches pointant vers la cellule).

Il existe quatre dépendances possibles pour une cellule LSTM, même si trois seulement peuvent exister en même temps :

— **L'état caché précédent dans le temps** : h_{t-1}. Si $t = 1$, cet état n'existe pas.
— **la cellule d'état précédente** : c_{t-1}. Une nouvelle fois si $t = 1$, cette cellule n'existe pas.
— **une observation au pas de temps actuel** : x_t. Cette dépendance n'existe que sur la première couche cachée.
— **L'état caché précédent dans la profondeur** : h_t^{l-1} . Cette dépendance existe pour toute couche cachée avec $l > 1$.

Ainsi, partant de la cellule c_t, on fournit l'information à la cellule c_{t+1} suivante et on calcule aussi l'état caché h_t. Et comme visible graphiquement, h_t a une influence potentielle sur la cellule LSTM c_{t+1}.

Le graphique précédent marque la différence entre un RNN et un réseau LSTM. En effet, avec un RNN, chaque état caché prend toute l'information provenant du passé et la transforme intégralement en lui appliquant une fonction. Une cellule LSTM, elle, prend l'information précédente et effectue quelques modifications (telles que des additions ou des multiplications).

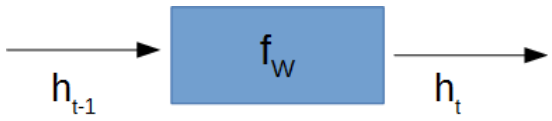


FIGURE 3.29 – Flux d'information avec un RNN

Le schéma précédent résume les limites d'un réseau de neurones récurrents. Durant la rétropropagation, les gradients ne peuvent pas « remonter le temps » facilement car les dérivées de la fonction *tanh* multipliées aux poids et sommées sont presque nulles. Le gradient est donc proche de 0.

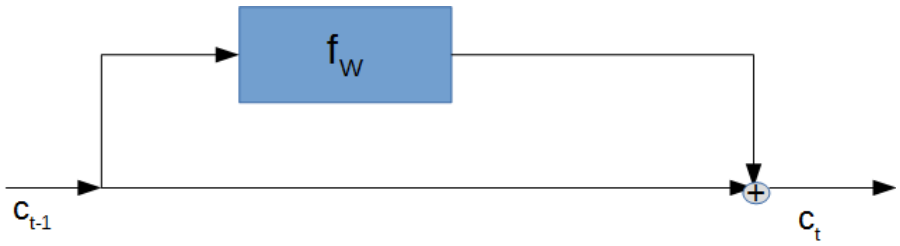


FIGURE 3.30 – flux d'information avec un LSTM

Ce schéma représente de manière simpliste le fonctionnement d'un LSTM. Cette vue a toute fois l'avantage d'être plutôt explicite. Ainsi, la valeur de la cellule d'état au temps précédent n'est pas transformée. On lui ajoute plutôt un autre vecteur. Et ce terme ajouté est une fonction de l'information précédente amenée par c_{t-1}. Cette fonction n'est toutefois pas la même que celle présente dans le cas d'un RNN et sera dévoilée dans la suite de la section.

De manière plus formelle on a donc que :

$$c_t = c_{t-1} + f_W(c_{t-1})$$

De manière plus développée :

$$c_t = c_1 + f_W(c_1) + f_W(c_2) + ... + f_W(c_{t-1})$$

Cette structure est la clé de la performance de l'approche LSTM sur l'approche RNN. En effet, lorsque l'on calcule la dérivée de l'expression précédente, cela va devenir une longue addition de dérivées de termes individuels. Les gradients sont toujours sommés ensemble, jamais multipliés. De cette manière on évite le problème de disparition du gradient et on facilite donc la remontée dans le temps.

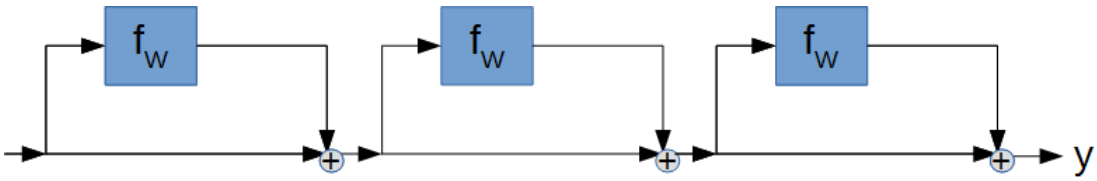


FIGURE 3.31 – Architecture LSTM déroulée

Ainsi, de manière plus formelle si :

$$y = f_1(x) + f_2(x) + f_3(x)$$

Alors

$$\frac{\partial y}{\partial x} = \frac{\partial f_1(x)}{\partial x} + \frac{\partial f_2(x)}{\partial x} + \frac{\partial f_3(x)}{\partial x}$$

Quelque soit la fonction f choisie (qui pourrait être notre fonction f_W par exemple), en utilisant la même fonction de manière répétée, la dérivée est simplement la somme de dérivées individuelles. Cette structure n'explosera ni ne s'annulera rapidement, ce qui était le but recherché. Et même si certains gradients s'annulent, ce n'est pas un problème. Ce n'est pas optimal mais puisque la dérivée repose sur des sommes, si quelques une d'entre elles sont nulles cela n'impliquera pas une nullité de la totalité du gradient.

Jusqu'ici, l'approche LSTM n'a pas été réellement explorée. C'était plus une présentation de l'avantage de l'approche. Toutefois, cet avantage semble avoir un biais évident. Si l'on se contente d'ajouter de l'information à chaque cellule d'état, cela ne fera que grandir et pourrait mener à une explosion du gradient sur de longues périodes temporelles.

Les cellules LSTM gèrent la mémoire de manière intelligente. Il est en effet possible de prendre en considération le contexte :

— Les cellules LSTM peuvent « oublier » (« resetting memory ») une information provenant de la cellule d'état précédent et qui n'est plus jugée utile.
— Les cellules LSTM peuvent « écrire » en mémoire une information provenant d'une nouvelle observation ou d'un état caché précédent.
— Les cellules LSTM peuvent également se contenter de « lire » une information contenue en mémoire.

Les possibilités décrites ci-avant correspondent aux concepts de « ré-initialisation de la mémoire » (« resetting memory »), « écriture en mémoire » (« writting to memory mémoire ») et « lecture de la mémoire » (« reading from memory »). Ii s'agit de principes très proches de ceux permettant par exemple le fonctionnement d'un système informatique. On décrit alors souvent une cellule LSTM par une « cellule mémoire ».

De manière plus précise, l'opération d'écriture en mémoire est additive. Il s'agit juste de rajouter une nouvelle information supposée pertinente au flux initial. L'opération de ré-initialisation de la mémoire est elle multiplicative et a lieu avant l'écriture en mémoire. Quand l'information provenant de la cellule précédente arrive, elle est multipliée pas un vecteur ayant des valeurs entre 0 et 1 pour effacer totalement ou retenir une partie de l'information supposée respectivement inutile ou pertinente. Enfin la lecture en mémoire est également une opération multiplicative avec un vecteur contentant des valeurs entre 0 et 1. Toutefois, cette partie ne modifie pas le vecteur en entrée. La lecture permet plutôt de déterminer quelle information pourra accéder à l'état caché et ainsi décider de ce qui l'influencera.

Les deux opérations multiplicatives le sont terme à terme :

$$\begin{bmatrix} a \\ c \end{bmatrix} \otimes \begin{bmatrix} b \\ d \end{bmatrix} = \begin{bmatrix} ab \\ cd \end{bmatrix}$$

Supposons par exemple que $\begin{bmatrix} a \\ c \end{bmatrix}$ soit le vecteur de ré-initialisation de la mémoire et $\begin{bmatrix} b \\ d \end{bmatrix}$ le vecteur d'information. Dans l'équation précédente, si a vaut 0 alors l'information contenue dans b est perdue. Si c par contre vaut 1 alors toute l'information de d sera conservée. Si par contre la valeur dans le vecteur de ré-initialisation est $0,5$ on peut considérer qu'il s'agit de réduire l'importance d'une entrée sans complètement l'effacer.

En utilisant les descriptions précédentes on a alors le graphique suivant :

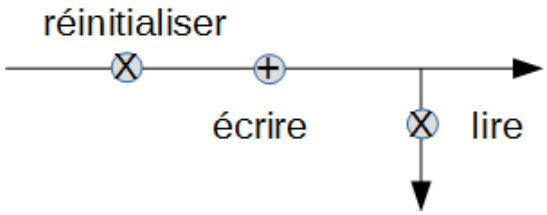


FIGURE 3.32 – Schéma simplifié d'une « cellule mémoire » LSTM

Le fonctionnement des cellules LSTM repose sur l'utilisation de cloisons (« gate » en anglais). En effet on cherche à cloisonner ce qui peut rentrer ou sortir d'une cellule LSTM. Ce que l'on utilise pour réinitialiser, écrire ou lire constituent des cloisons.

Il en existe quatre :

— **f** : la « cloison d'oubli » (ou « *forget gate* » en anglais). C'est cet outils qui élimine, diminue ou retient l'information provenant d'une cellule d'état précédente. C'est la première interaction réalisée et c'est une opération multiplicative. La fonction sigmoïde est utilisée par cette opération qui retourne, pour rappel, des valeurs entre 0 et 1. Quand la valeur de cette cloison vaut 0, l'algorithme oublie. Quand la valeur est de 1, l'approche retient l'intégralité de l'information.

— **g** Cette cloison n'a pas de nom. Toutefois, elle est en partie en charge de l'opération d'écriture. Elle stocke une valeur entre -1 et 1 qui traduit à quel point une information devrait être ajoutée à la cellule d'état. Il s'agit de la valeur en entrée de la cellule d'état. Ce calcul est effectué avec une fonction tangente hyperbolique.

— **i** : la cloison d'entrée (ou « *input gate* » en anglais). Il s'agit de l'autre cloison en charge de l'écriture en mémoire. Elle contrôle la proportion de g qui devrait être autorisée à rentrer. Il s'agit donc d'une valeur entre 0 et 1, calculée à l'aide de la fonction sigmoïde.Dans la mesure où elle bloque une information en entrée, cette cloison est similaire à la cloison d'oubli. Dans la pratique, on multiplie i et g et on ajoute le résultat à la cellule d'état. Puisque i est compris entre 0 et 1, et g entre -1 et 1, la valeur ajoutée est comprise entre -1 et 1.

— **o** : la « cloison de sortie » (ou « *output gate* » en anglais). La valeur retournée est aussi calculée à l'aide d'une sigmoïde. Cette cloison détermine quelle valeur sortie de la cellule d'état. Cette cloison permet la lecture de la mémoire. Enfin, en multipliant la valeur de cette cloison avec la transformée par la fonction *tanh* de la cellule d'état on obtient la nouvelle valeur de l'état caché.

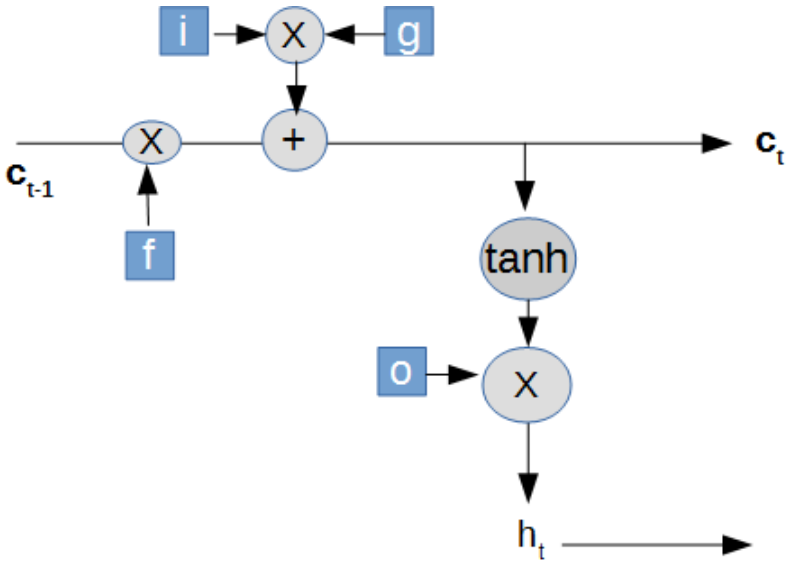


FIGURE 3.33 – Schéma d'une « cellule mémoire » LSTM

Le graphique ci-dessus reprend visuellement les éléments décrits précédemment.

Mathématiquement on obtient :

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot tanh(c_t)$$

où $\odot$ est le produit terme terme

On peut généraliser la première équation à une architecture à plusieurs couches cachées. On aurait alors :

$$c_t^l = f^l \odot c_{t-1}^l + i^l \odot g^l$$

Comme l'équation précédente le montre, la cellule d'état n'a pas de fonction d'activation. Pour autant, la cellule n'explose pas grâce aux processus d'oubli et d'écriture.

En outre, les cloisons sont définies de la façon suivante :

$$f_t = sigmoïde(W_{x,f}x_t + W_{h,f}h_{t-1} + b_f)$$
$$g_t = tanh(W_{x,g}x_t + W_{h,g}h_{t-1} + b_g)$$
$$f_t = sigmoïde(W_{x,i}x_t + W_{h,i}h_{t-1} + b_i)$$
$$f_t = sigmoïde(W_{x,o}x_t + W_{h,o}h_{t-1} + b_o)$$

Tout comme pour les états cachés d'un RNN, l'indice de chaque matrice de poids décrit l'utilité de la fonction. Par exemple, $W_{x,f}$ sont les poids qui lient l'entrée x à la cloison d'oubli f.

Toutes les cloisons présentées dans les équations précédentes ont leur propre poids et sont fonction du pas de temps du dernier état caché et de toute information disponible au pas de temps actuel. Cela paraît intuitif. En effet, les cloisons aident à modifier la cellule d'état en fonction du contexte actuel. Et le contexte est constitué des entrées au pas de temps actuel et des états cachés au temps précédent. La pratique semble donc en accord avec la théorie.

L'atout majeur de l'approche LSTM est que tous les processus sont différentiables. La structure permet de choisir quelles données utiliser et dans quelle proportion en fonction du contexte (pour rappel, le contexte est composé par les états cachés et les entrées). L'algorithme apprend ce schéma au cours de son apprentissage.

Enfin, les fonctions utilisées pour les portes correspondent à l'emploi le plus courant. Toutefois, il est tout à fait possible d'utiliser d'autres fonctions d'activation.

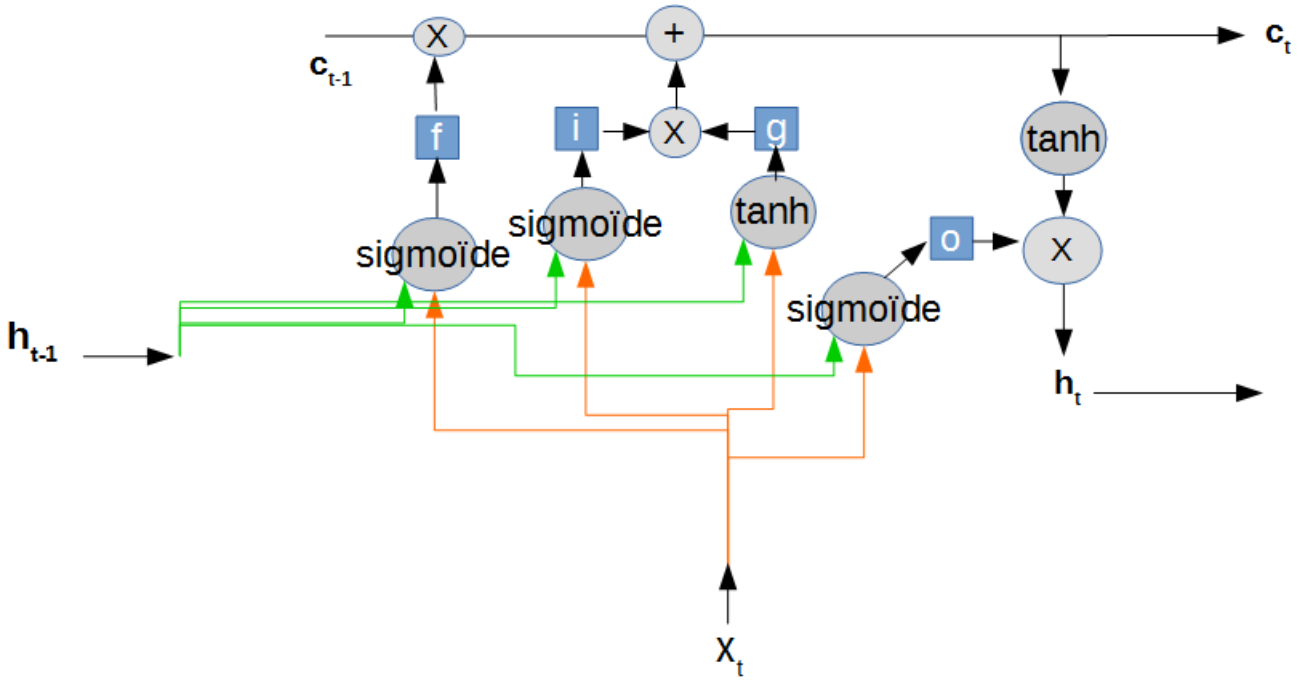


FIGURE 3.34 – Schéma avancée d'une « cellule mémoire » LSTM

Le schéma précédent donne une vue complète de ce qu'est une cellule mémoire LSTM. Puisqu'il existe une entrée x_t, l'hypothèse tacite est que l'on est sur la première couche cachée à un pas de temps t ($t > 1$) quelconque.

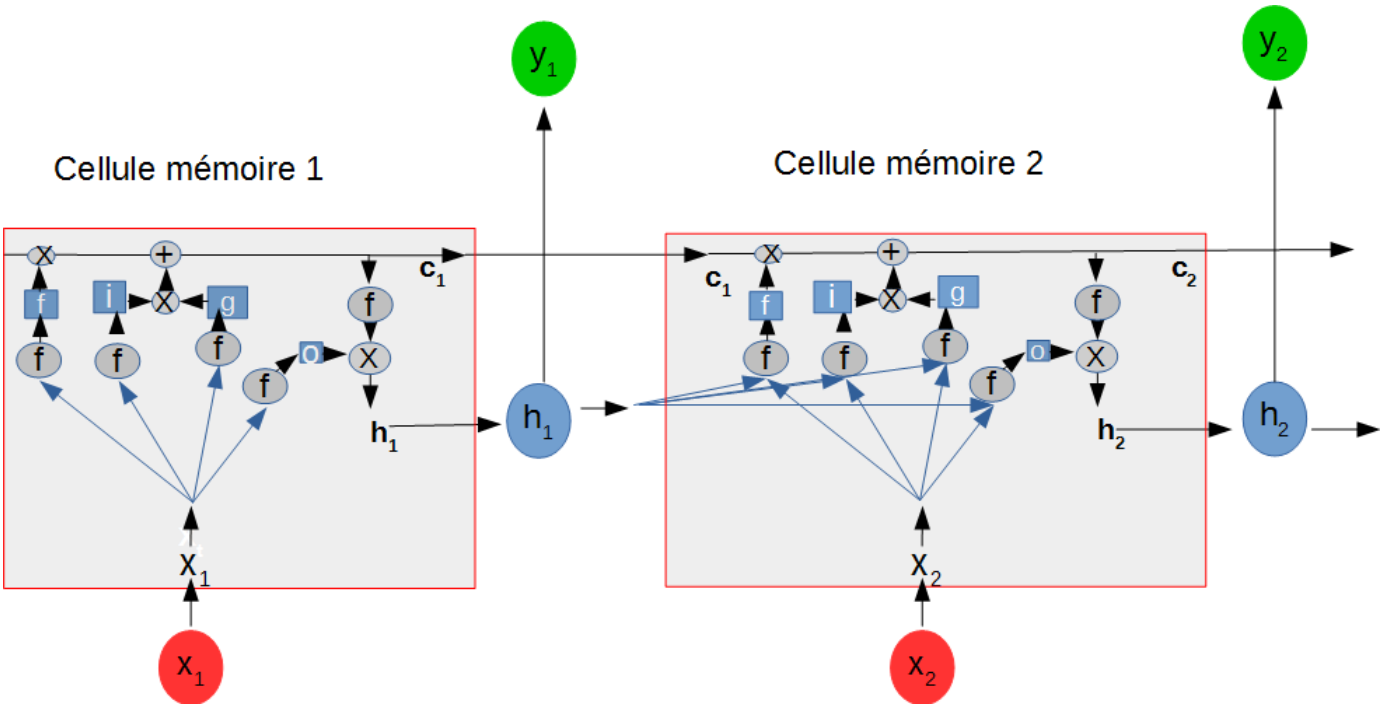


FIGURE 3.35 – Schéma d'un réseau LSTM déroulé avec f fonction d'activation quelconque

3.3.5 Analyse des résultats

Traitement de la variable cible

La charge totale qui est notre variable cible est marquée par un pic important de valeurs très proche de 0. Or, l'approche LSTM est très sensible au signal qu'il doit prédire. Ainsi, afin de se ramener à une information plus simple à lire pour l'algorithme, une transformation de la charge totale à l'aide de la fonction logarithme a été effectuée. Plus précisément, si y est la variable cible originale alors on a :

$$\tilde{y} = log(y + 1)$$

où $\tilde{y}$ est le nouveau signal à prédire.

Une fois la prédiction effectuée, afin de retrouver la valeur originale, il suffira d'opérer les transformations inverses :

$$y = e^{\tilde{y}} - 1$$

Comparaison aux données réelles

Afin de modéliser notre problématique, c'est un LSTM à une couche cachée qui a été utilisé. Après optimisation du nombre de neurones à l'aide de l'optimisation bayésienne, c'est une couche à 50 neurones qui a été utilisée. Les autres paramètres ont conservé leur valeur par défaut.

Le tableau suivant compare la provision globale estimée par l'approche LSTM à la provision réelle observée à juin 2017.

Provision observée	Provision estimée avec LSTM
-4 845 953 €	-2 425 484 €

montants en euros

FIGURE 3.36 – Comparaison des provisions estimées (avec LSTM) et des provisions réelles sur l'échantillon test

On constate que l'approche neuronale est pessimiste car elle sous-évalue la baisse de la charge sur la période de plus de 2 millions d'euros (soit un écart relatif par rapport à la provision réelle de presque 50%).

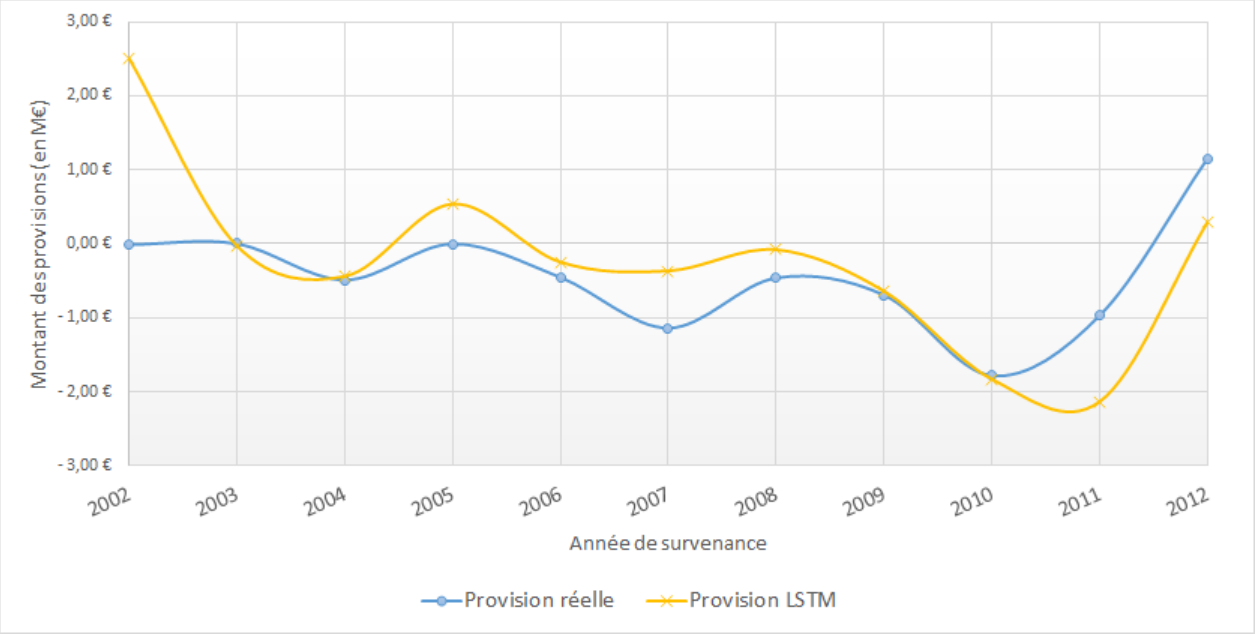


FIGURE 3.37 – Analyse de l'erreur globale d'estimation avec XGBoost

Quand on analyse de manière plus détaillée les estimations de provision de l'approche LSTM sur le graphique précédent, on constate deux grandes particularités.

Tout d'abord, au global, la tendance suivie par la charge réelle observée est relativement bien imitée par l'algorithme LSTM. Cela se traduite par deux courbes très proches graphiquement et mêmes confondues certaines années comme en 2004, 2009 et 2010.

Le second élément d'analyse est le décrochage de l'estimation sur quelques années en particulier. Ainsi, un écart très grand sépare les points pour l'année de survenance 2002. Des écarts importants (i.e. de plus de 500 000 euros) existent également pour les années de survenance 2005, 2007 et 2011.

Enfin, un dernier point remarquable est que mise à part pour les années les plus récentes (i.e. les années postérieures à 2011), l'algorithme LSTM a tendance à surestimer la provision réelle observée (graphiquement les points en jaune sont plus souvent au dessus des points en bleu).

Comparaison aux autres méthodes

Cette section résume les résultats des estimations de provision obtenus à l'aide des trois méthodes testées : Chain-Ladder, LSTM et XGBoost.

Année de survenance	Provision réelle	Provision CL	Provision XGBoost	Provision LSTM
2002	-11 970 €	-8 487 €	339 870 €	2 507 240 €
2003	0 €	-69 830 €	-164 520 €	-26 563 €
2004	-488 232 €	-134 225 €	-314 675 €	-439 624 €
2005	-1 652 €	-79 148 €	24 773 €	538 649 €
2006	-454 292,00 €	-336 112 €	-596 011 €	-248 235 €
2007	-1 138 121 €	-564 787 €	-1 071 289 €	-366 879 €
2008	-461 045 €	-300 691 €	-39 057 €	-79 211 €
2009	-692 705 €	-392 928 €	-445 495 €	-641 277 €
2010	-1 782 887 €	-270 554 €	-1 073 863 €	-1 829 730 €
2011	-969 772 €	330 231 €	-1 655 465 €	-2 134 382 €
2012	1 154 722 €	1 077 049 €	-15 950 €	294 528 €
Total	**-4 845 953 €**	**-749 482 €**	**-5 011 681 €**	**-2 425 484 €**

montants en euros

FIGURE 3.38 – Analyse de l'erreur d'estimation des provisions avec Chain-Ladder, XGBoost et LSTM

Sur l'estimation de la provision globale, c'est l'approche à base d'arbres XGBoost qui est la plus performante. En effet, c'est elle qui est la plus proche de la provision globale réelle observée à posteriori. L'erreur commise par XGBoost est d'un peu plus de 3% (soit 300 000 euros environ). Ensuite, c'est l'algorithme LSTM qui est le plus proche de la valeur observée avec un écart de plus de 2,4 millions d'euros tout de même. C'est donc l'approche classique par Chain-Ladder qui est la plus éloignée du résultat global (plus de 4 millions d'euros de différence).

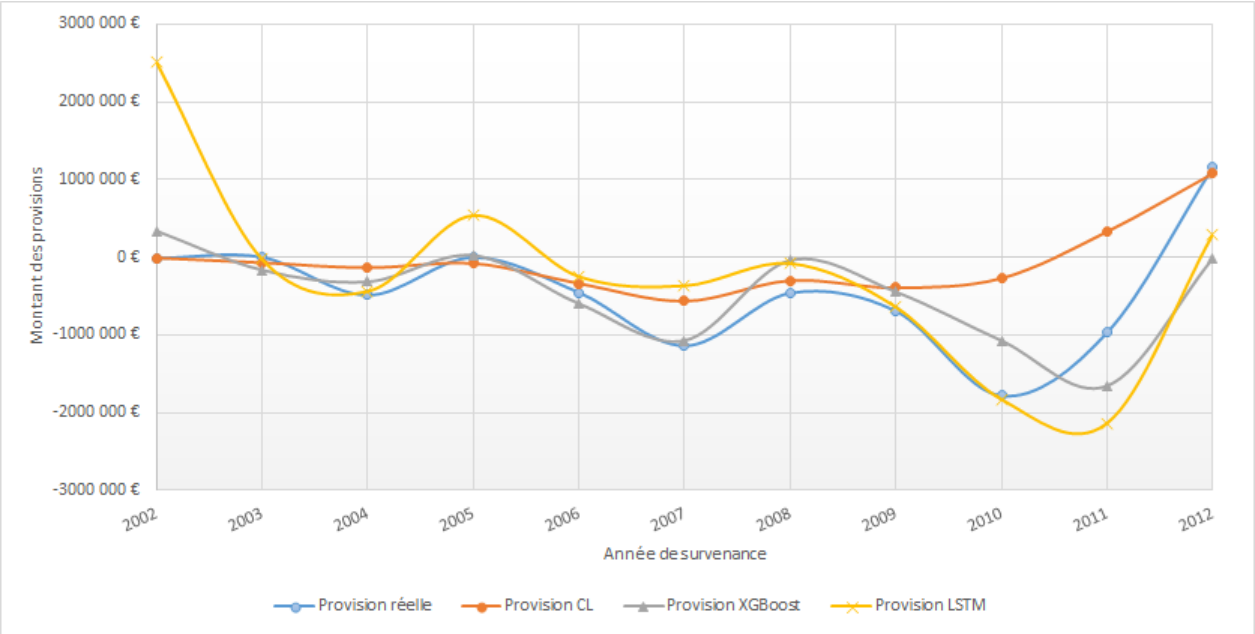


FIGURE 3.39 – Comparaison de l'estimation des charges totales avec Chain-Ladder, XGBoost et LSTM

Graphiquement on constate que Chain-Ladder est très proche de la réalité pour l'année de survenance 2002. Les deux nouvelles méthodes n'ont pas capté la tendance réelle qui était alors à la baisse (-11 970 euros) et ont commis une erreur significative dans l'estimation du montant de la provision (avec 350 000 euros de différence au minimum avec la valeur réelle observée).

Ensuite, jusqu'en 2009, même si la méthode de Chain-Ladder estime légèrement moins bien les tendances réelles que les méthodes XGBoost et LSTM, les quatre courbes restent pour autant assez proches. La véritable différence entre les nouvelles techniques et Chain-Ladder est visible sur les dates de survenance les plus récentes (i.e. à partir de 2010) où les écarts entre l'approche classique et les provisions observées explosent. Les méthodes XGBoost de LSTM s'en sortent mieux en collant plus fidèlement aux provisions réellement observées.

Le propos précédent est toutefois à nuancer car pour les sinistres avec une année de survenance en 2012, c'est bien Chain-Ladder qui a l'estimation la plus proche de la réalité observée.

Conclusion sur l'utilisation de l'algorithme LSTM

Même si la technique neuronale s'est montrée sans doute moins performante que XGBoost, elle n'en reste pas moins un outil formidable. En effet, grâce à caractère modulable, et par sa construction ingénieuse, elle est capable de s'adapter à pratiquement tous les problèmes, qu'ils soient séquentiels ou non. C'est pour cette raison que l'approche a pu fournir des résultats encourageants pour son application à un domaine sans doute peu habituel pour ce genre de technique.

Les résultats moins bons que ceux produits par l'approche XGBoost par exemple sont en partie dûs à sa limite principale : sa structure extrêmement complexe. C'est cette architecture qui rend la technique théoriquement très efficace. Dans la pratique, des contraintes opérationnelles arrivent très vite. L'optimisation du très grand nombre de paramètres de l'algorithme (les poids reliant chaque couche du réseau) requiert un temps de calcul très vite très important. Dans le même ordre d'idée, une mémoire de grande taille est nécessaire si l'on souhaite pour effecteur les calculs liés à l'algorithme tout en utilisant son ordinateur de manière habituelle.

Enfin, l'approche LSTM est une boîte noire de bout en bout. Compte tenu des différentes transformations effectuées tout au long de la structure neuronale, il serait extrêmement complexe de déterminer le rôle exact des variables utilisées sur le résultat final.

En définitive, les résultats fournis par l'algorithme LSTM sont prometteurs. Ils sont d'autant plus prometteurs que l'optimisation de l'approche n'est pas aussi aboutie que celle pour la l'algorithme XGBoost. Cela en grande partie pour des raisons opérationnelles et les difficultés à utiliser ce genre de méthode sur des ordinateurs standards. C'est là que se trouve la principale limite de cette méthode.

Chapitre 4

Contexte réglementaire

4.1 Introduction générale

L'IASB (pour International Accounting Standards Board, appelé le Board par la suite) a publié en mai 2017, IFRS 17 Insurance Contracts ([6]). Il s'agit du premier vrai standard international IFRS concernant les contrats d'assurance. Il devrait permettre aux investisseurs et autres acteurs du monde financier de mieux comprendre l'exposition aux risques des assureurs, leur rentabilité et leur situation financière.

La norme IFRS 17 remplacera IFRS 4, norme introduite comme Standard par intérim en 2004. La norme IFRS 4 laissait la possibilité aux différentes compagnies de produire leurs états financiers en utilisant les normes comptables locales. Cela a entraîné une multitude d'approches différentes. À cause de cette pluralité des normes, il est difficile pour les investisseurs de pouvoir comparer les performances financières de compagnies aux activités pourtant similaires (ici il s'agit de compagnies d'assurance).

La norme IFRS 17 a pour but de résoudre les problèmes de comparabilité inhérents à la norme IFRS 4. Pour cela, elle imposera à tous les contrats d'assurance d'être comptabilisés de manière identique quelque soit la localisation de la société ayant émis ce contrat.

L'une des principales innovations de la norme IFRS 17 est l'utilisation de valeurs actuelles dans les calculs au lieu des coûts historiques. L'information sera donc mise à jour de manière régulière, fournissant de précieuses informations aux utilisateurs des états financiers publiés par les compagnies.

La norme IFRS 17 s'applique aux contrats d'assurance. Ainsi, la norme IFRS 17 s'applique donc à toute compagnie proposant des contrats d'assurance, que l'activité principale de la société soit l'assurance ou non. Toutefois, la plupart des contrats sont émis par des compagnies d'assurance.

Enfin, la norme IFRS 17 sera effective dès le premier janvier 2021. Cependant, une société pourra décider d'appliquer cette norme avant cette date si, et seulement si, elle applique également les normes IFRS 9 (« Financial Instruments ») et IFRS 15 (« Revenue from Contracts with Customers »).

4.2 Les contrats d'assurance

L'IASB définit un contrat d'assurance (ou police d'assurance) de la manière suivante (paragraphe IN6, sous-paragraphe a) :

> « The key principles in IFRS 17 are that an entity :
>
> · identifies as insurance contracts those contracts under which the entity accepts significant **insurance risk** from another party (the policyholder) by agreeing to compensate the policyholder if a specified uncertain future event (the insured event) adversely affects the policyholder. »

Un risque d'assurance est un risque financier qui n'empêche pas la définition d'un contrat comme étant un contrat d'assurance. Parmi les pré-requis pour être un risque d'assurance il est nécessaire que :

— Le risque pré-existe, i.e., le risque ne doit pas résulter uniquement de la création du contrat
— Le risque doit affecter négativement l'assuré

De plus, l'annexe B paragraphe B18 de la norme IFRS 17 donne une définition du terme significatif :

> « Insurance risk is significant if, and only if, an insured event could cause an insurer to **pay significant additional benefits in any scenario,** excluding scenarios that have no commercial substance (i.e. have no discernible effect on the economics of the transaction). If significant additional benefits would be payable in scenarios that have commercial substance, the condition in the previous sentence may be met even if the insured event is extremely unlikely or even if the expected (i.e. probability-weighted) present value of contingent cash flows is a small proportion of the expected present value of all the remaining contractual cash flows. »

La définition d'un risque d'assurance significatif est donc relativement floue puisque selon la définition précédente, il s'agit d'un risque qui puisse entraîner le paiement d'avantages supplémentaires significatifs dans n'importe quel scénario. Chacun pourra donc interpréter à sa manière ce qu'est le paiement d'un avantage supplémentaire significatif. Un biais d'interprétation est donc introduit dès la définition d'un contrat d'assurance.

Il existe toutefois des contrats qui pourraient être traités comme des contrats d'assurance mais qui sont pourtant exemptés (section **Scope**) :

> «
> (a) warranties provided by a manufacturer, dealer or retailer in connection with the sale of its goods or services to a customer (see IFRS 15 *Revenue from Contracts with Customers*[1]).
> (b) employers' assets and liabilities from employee benefit plans (see IAS 19 *Employee Benefits* and IFRS 2 *Share-based Payment*) and retirement benefit obligations reported by defined benefit retirement plans (see IAS 26 *Accounting and Reporting by Retirement Benefit Plans*).

1. reference

(c) contractual rights or contractual obligations contingent on the future use of, or the right to use, a non-financial item (for example, some licence fees, royalties, variable and other contingent lease payments and similar items : see IFRS 15, IAS 38 *Intangible Assets* and IFRS 16 *Leases*).

»

Les exemptions proposées existent lorsque des conflits apparaissent dans la comptabilisation de certains contrats. Ainsi, pour éviter que des contrats ne soient comptabilisés de plusieurs façons différentes au sein d'un même groupement de normes, il a donc été important de préciser la portée d'application de la norme IFRS 17. Parmi ces contrats exemptés se trouvent les garanties d'assurance proposées par des entreprises en lien avec la vente de biens ou de services aux clients (par exemple la garantie en cas de casse lors de l'achat d'un téléphone mobile). Sont également hors de portée de la norme IFRS 17 les produits liés aux engagements sociaux et les contrats concernant l'utilisation future ou actuelle de biens non financiers (cette clause exclue les contrats sur les droits de licence par exemple).

Les conséquences financières d'un contrat d'assurance ne sont pas connues avant plusieurs années du point de vue de l'assureur. La résultante d'un contrat peut, pour un assureur, être un profit ou une perte. Ainsi, comptabiliser un contrat d'assurance repose essentiellement sur les hypothèses prises en compte. C'est donc une vraie problématique d'essayer de mesurer et rapporter de manière fiable la performance des contrats d'assurance.

Les pratiques utilisées par les compagnies d'assurance ont évolué au cours du temps, souvent en fonction du contexte dans le pays dans lesquelles elles se trouvent. Et, dans de nombreux cas, les modèles de comptabilité utilisés par les compagnies d'assurance sont même inconsistants avec les standards IFRS employés par d'autres industries du même pays. Les possibilités de comparaison avec les autres secteurs de l'industrie sont donc limitées.

Pour faire face à ces problématiques, les bénéfices de la norme IFRS 17 devraientalors être les suivantes :

— Comparabilité des sociétés quelque soit le pays : les assurances appliqueront des méthodes de comptabilisation consistantes pour les contrats d'assurance
— Comparabilité entre les contrats d'assurance : une compagnie internationale utilisera une même méthode de comptabilisation des contrats d'assurance à l'échelle du groupe. Cela permettra de simplifier la comparaison des résultats par produit et par zone géographie
— Comparabilité au sein des différentes industries : les revenus refléteront l'activité d'assurance pure

4.3 Agrégation des contrats d'assurance

Une fois que l'on a pu déterminer les contrats pour lesquels il faut appliquer la norme IFRS 17, il faut encore regrouper ces contrats en sous catégories.

La section « Level of aggregation of insurance contracts » définit la manière dont doivent être construits les groupes.

«

14 An entity shall identify portfolios of insurance contracts. A portfolio comprises contracts subject to similar risks and managed together. Contracts within a product line would be expected to have similar risks and hence would be expected to be in the same portfolio if they are managed together. Contracts in different product lines (for example single premium fixed annuities compared with regular term life assurance) would not be expected to have similar risks and hence would be expected to be in different portfolios.

»

Ainsi la première maille de regroupement consiste à trouver les contrats ayant des risques similaires. Comme annoncé par la norme, les contrats d'une même ligne de produits devraient ainsi avoir une certaine homogénéité dans les risques couverts et donc être dans un même groupe. Il s'agit donc de créer des portefeuilles de contrats.

«

22 An entity shall not include contracts issued more than one year apart in the same group. To achieve this the entity shall, if necessary, further divide the groups described in paragraphs 16–21.

»

Une fois que des groupes homogènes ont été constitués, il est nécessaire les regrouper par cohorte. Ainsi, ne peuvent être dans un même groupe des contrats avec des dates d'émission séparées de plus d'un an. Il s'agit donc de construire des cohortes annuelles.

«

16 An entity shall divide a portfolio of insurance contracts issued into a minimum of :

(a) a group of contracts that are onerous at initial recognition, if any ;

(b) a group of contracts that at initial recognition have no significant possibility of becoming onerous subsequently, if any ; and

(c) a group of the remaining contracts in the portfolio, if any..

»

Enfin, au sein de chaque cohorte annuelle, il est nécessaire de regrouper les contrats en trois sous groupes potentiels : des contrats onéreux, des contrats profitables sans grande possibilité de devenir onéreux et enfin d'autres contrats.

Ainsi avant même de comptabiliser les contrats, un travail très conséquent de segmentation est nécessaire. Cette segmentation repose sur un concept clé, les contrats onéreux. Dans la section « Measurement », partie « Onerous contracts » paragraphe 47, on a la définition suivante :

«

47 An insurance contract is onerous at the date of initial recognition if the fulfilment cash flows allocated to the contract, any previously recognised acquisition cash flows and any cash flows arising from the contract at the date of initial recognition in total are a net outflow. Applying paragraph 16(a), an entity shall group such contracts separately from contracts that are not onerous. To the extent that paragraph 17 applies, an entity may identify the group of onerous contracts by measuring a set

of contracts rather than individual contracts. An entity shall recognise a loss in profit or loss for the net outflow for the group of onerous contracts, resulting in the carrying amount of the liability for the group being equal to the fulfilment cash flows and the contractual service margin of the group being zero.

»

Ainsi, un contrat d'assurance est onéreux à la date de la comptabilisation initiale si les flux de trésorerie liés à l'exécution du contrat, les flux de trésorerie d'acquisition précédemment comptabilisés et les flux de trésorerie découlant du contrat à la date de la comptabilisation initiale constituent au total une sortie nette pour l'entreprise. Cette perte doit alors être comptabilisée directement dans les pertes et profits de l'entreprise.

La manière dont l'entreprise relie des flux de trésorerie à un contrat, qui peut varier d'une entreprise à l'autre, détermine le groupe dans lequel sera classé le contrat. Il peut donc se poser la question de la comparabilité des groupes construits par chaque entreprise.

Le schéma suivant résume les différents groupes imposés par la nouvelle norme.

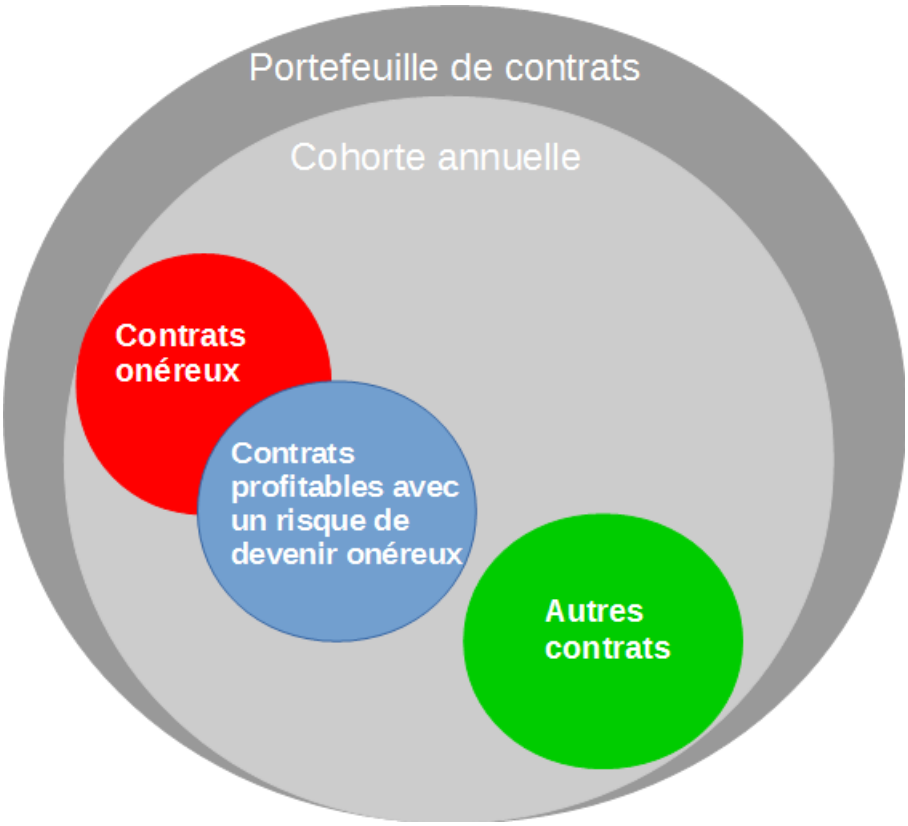


FIGURE 4.1 – Segmentation des contrats avec IFRS 17

4.4 Le bilan prudentiel sous IFRS 17

La norme IFRS 17 impose aux compagnies qui proposent des contrats d'assurance de les comptabiliser dans leur bilan en les décomposant ainsi :

— La **valeur actuelle des prestations ou fulfilment cash flows** (section « Estimates of future cash flows », paragraphes B36 à B71) : la valeur actuelle de l'estimation des montants que l'assureur s'attend à collecter via les primes, et ceux qu'elle s'attend à payer via ses charges. Ce montant comprend un ajustement lié à l'incertitude des ces flux de trésorerie.

— La **marge de service contractuelle ou Contractual Service Margin** (section « Contractual service margin », paragraphe 38) : le futur profit attendu suite à la couverture d'assurance fournie (i.e. le bénéfice non gagné)

De manière schématique on a alors :

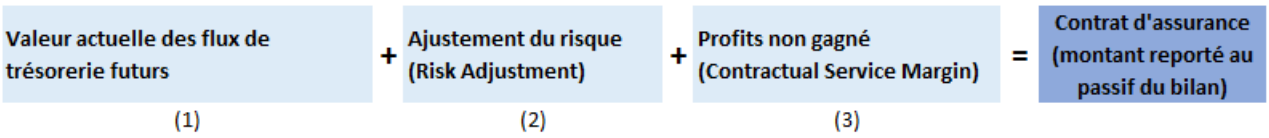


FIGURE 4.2 – Modèle IFRS 17

L'ajustement du risque (noté RA par la suite) est définie comme suit par la norme comme suit (section « Risk adjustment for non-financial risk (paragraphs B86–B92) » :

> «
> **37** An entity shall adjust the estimate of the present value of the future cash flows to reflect the compensation that the entity requires for bearing the uncertainty about the amount and timing of the cash flows that arises from non-financial risk.
> »

La RA est donc un montant reflétant l'incertitude concernant les flux de trésorerie qui constituent le bilan. Enfin, les blocs (1) et (2) du graphique précédent définissent le fulfilment cash flows.

Afin de refléter au mieux les changements liés aux contrats d'assurance et aux risques liés à ceux-ci, la norme IFRS 17 imposera aux compagnies de mettre à jour les hypothèses utilisées à chaque date de clôture afin que celles-ci soient en accord avec les informations disponibles sur les marchés. Les changements dans la comptabilisation des contrats d'assurance dus au contexte économique (comme un changement dans les taux d'intérêt) seront disponibles dans les états financiers publiés par la compagnie.

La comparaison entre les normes IFS 4 et IFRS 17 est présentée dans le tableau suivant.

Exemple de profits et pertes sous IFRS 4				Exemple de profits et pertes sous IFRS 17			
	Année 1	Année 2	Total		Année 1	Année 2	Total
Primes	10 000	-	10 000	Revenu de l'assurance	360	250	610
Charge de sinistres et autres dépenses	-1 000	-11 100	-12 100	Charge de sinistres et autres dépenses	-10	-	-10
Changement dans le passif du contrat d'assurance	-10 000	10 000	0	**Résultat de l'assurance**	350	250	600
Revenu d'investissement	1 500	1 200	2 700	Revenu d'investissement	1 500	1 200	2 700
Profit ou perte	**500**	**100**	**600**	Frais financiers liés à l'assurance	-1 500	-1 200	-2 700
				Résultat financier net	-	-	-
en unité arbitraire (UA)				**Profit ou perte**	**350**	**250**	**600**

FIGURE 4.3 – Comparaison entre IFRS 4 et IFRS 17

La partie gauche de l'exemple précédent illustre une manière commune de présenter le résultat pour un groupe de contrats d'assurance sous IFRS 4. Toutefois, compte tenu dans la grande diversité des pratiques de comptabilisation sous IFRS 4, la représentation précédente n'est pas nécessairement représentative d'une pratique spécifique (au niveau d'une compagnie ou d'une zone géographique).

Le revenu généré par ce groupe de contrat d'assurance correspond aux primes acquises (10 000UA), plus la valeur des investissements réalisés grâce aux primes (2 700UA) moins les charges et les dépôts (12 100UA).

Cet exemple a pour but d'illustrer deux changements significatifs pour un groupe de contrats d'assurance. En particulier, avec la norme IFRS 17 :

— la présentation des primes en tant que revenus et dépenses (au sein de la ligne « changement dans le passif du contrat d'assurance ») est retirée.
— une compagnie d'assurance peut présenter deux indicateurs de performance financière séparément avec le « résultat de l'assurance » et le « résultat financier net ».Cela permet d'expliquer la rentabilité des contrats d'assurance.

Ainsi la lisibilité des résultats est simplifier à l'aide de l'approche IFRS 17.

4.5 Éléments d'information sur l'apport de nouvelles méthodes de provisionnement au cadre défini par IFRS 17

Dans le cas de cette nouvelle norme, les méthodes de projection à l'ultime pourrait avoir un rôle à jouer. Et notamment dans la construction des groupes de contrats.

Dans la section « Onerous contracts » paragraphe 48, il est décrit comment un groupe de contrats peut devenir onéreux ou encore plus onéreux.

«

48 A group of insurance contracts becomes onerous (or more onerous) on subsequent measurement if the following amounts exceed the carrying amount of the contractual service margin :

(a) unfavourable changes in the fulfilment cash flows allocated to the group arising from changes in estimates of future cash flows relating to future service ; and

(b) for a group of insurance contracts with direct participation features, the entity's share of a decrease in the fair value of the underlying items.

Applying paragraphs 44(c)(i), 45(b)(ii) and 45(c)(ii), an entity shall recognise a loss in profit or loss to the extent of that excess.

»

En tant normal, un groupe de contrats d'assurance soumis à IFRS 17 est déclaré onéreux ou non dès la création de celui-ci. Toutefois, en fonction des aléas de la vie des contrats, ce groupe peut devenir onéreux. Pour cela, il est nécessaire (d'après le sous paragraphe a) qu'apparaissent des modifications défavorables des flux de trésorerie futurs associées à ce groupe de contrats. Ces modifications doivent être dues à des variations des services d'assurance futurs associées à ce groupe. Cela peut être le cas lorsque des sinistres sont déclarés. En fonction de l'importance de ces sinistres et surtout s'ils dépassent le montant de la CSM initialement prévue pour ce groupe de contrats, alors le groupe devient onéreux.

Or, les algorithmes XGBoost et LSTM ont cette capacité de pouvoir proposer une estimation relativement fiable de ce que sera la charge ultime associée à un sinistre. Ainsi, en comparant ces estimations à la CSM, il sera possible d'évaluer si les contrats changent de catégorie ou non lorsque des sinistres sont déclarés. Connaître ces catégories est un véritable enjeu car une fois déclaré comme onéreux, les pertes associés à un groupe de contrats sont immédiatement répercutées sur le résultat de l'entreprise.

Toutefois, avant de pouvoir comparer les estimations de charge totale à la CSM, il faut retraiter les potentiels effets monétaires. Pour cela, des hypothèses sont nécessaires (concernant l'inflation jusqu'à l'ultime par exemple). De la même façon des hypothèses supplémentaires sur l'actualisation de ces flux pourraient également être nécessaires. L'utilisation des estimations de charge ultime obtenues via LSTM ou XGBoost ne serait donc pas immédiate.

Enfin, comme présenté dans la section précédente, toute estimation de la valeur actuelle de flux futurs doit être accompagnée d'un montant d'ajustement du risque traduisant l'incertitude concernant ces flux.

«

119 An entity shall disclose the confidence level used to determine the risk adjustment for non-financial risk. If the entity uses a technique other than the confidence level technique for determining the risk adjustment for non-financial risk, it shall disclose the technique used and the confidence level corresponding to the results of that technique.

»

Ce montant de RA doit être basé sur un intervalle de confiance. Obtenir un intervalle de confiance des résultats pour les approches XGBoost et LSTM est quelque chose de théoriquement assez simple. Il suffit de rééchantillonner la base d'apprentissage et d'analyser la distribution des résultats. On aura alors une idée de la marge d'erreur à prendre lorsque l'on utilise les estimations. Dans la pratique, pour que l'intervalle de confiance soit fiable, il faut utiliser un très grand nombre de points. Très vite une problématique de temps de calcul apparaîtrait alors. Toutefois, avec un matériel informatique adéquat, cette problématique est loin d'être insurmontable.

Ainsi, l'utilisation de ces nouvelles méthodes peut s'inscrire parfaitement dans le cadre prévu par IFRS 17.

Conclusion

Au cours de ce mémoire ont été présentées deux approches innovantes qui avaient pour objectif d'améliorer les méthodes de provisionnement classiques, de type Chain-Ladder. Ces nouvelles approches devaient permettre de mieux prendre en compte l'information disponible sur un sinistre et ainsi aider à obtenir des estimations de provision plus fiables, moins marquées par le jugement d'expert d'une part. D'autre part, ces nouvelles techniques devaient permettre l'individualisation et donc le suivi à une échelle très fine du coût de chaque sinistre. Dans un premier temps, une grande importance a été donné à l'étude des données. Il s'agissait du portefeuille des Garanties contre les Accidents de la Vie d'un bancassureur français. Cette analyse a permis de mettre en avant une particularité du portefeuille qui était de contenir un très grand nombre de sinistres déclarés sans suite ou non garanti. Puis, deux modèles ont été calibrés sur un historique de sinistres allant de 2002 à fin juin 2012. La modélisation choisie était basée sur une projection de la charge totale de chaque sinistre directement jusqu'à l'ultime. L'horizon de temps supposée pour accéder à l'ultime étant de 5 ans. Le premier modèle testé est une régression à l'aide d'un Gradient Boosting d'arbres de décision : XGBoost. Le second modèle mis en place est l'algorithme Long Short-Term Memory (encore appelé LSTM) qui repose sur les réseaux de neurones. Chaque approche a permis d'avoir, pour chaque sinistre d'un échantillon de test, une estimation du montant qu'il aurait fallu provisionner vue à aujourd'hui. Ces estimations ont été comparées aux provisions réelles observées à aujourd'hui et aux provisions estimées par Chain-Ladder. Dans la pratique, les deux nouvelles méthodes se sont montrés capables d'avoir une estimation globale du montant de provision meilleure que Chain-Ladder. De manière plus fine, les deux nouvelles approches se sont réellement démarquées de Chain-Ladder pour les estimations de provision des sinistres avec les années de survenance les plus récentes. Les méthodes XGBoost et LSTM ayant alors des estimations de provision bien plus proches des montants réels que Chain-Ladder. C'est l'algorithme XGBoost qui s'est révélé avoir les meilleurs résultats parmi les deux méthodes testées. Enfin, les deux méthodes présentées au cours de cette étude pourraient être de vrais atouts dans le cadre de l'utilisation par une entreprise de la norme IFRS 17. Elles permettraient de déterminer si un groupe de contrats peut ou est devenu onéreux. Cela aurait alors un impact direct sur le résultat de la compagnie.

Les méthodes utilisées ont permis d'avoir des résultats satisfaisants, mais il faut toutefois relativiser les remarques précédentes. Tout au long du mémoire, les multiples limites auxquelles nous avons dû faire face ont été listées. Ces contraintes étaient principalement dues au traitement de la base de données. En effet, il a été nécessaire à de nombreuses reprises d'émettre des hypothèses qui ont sans doute biaiser les résultats. Ces hypothèses portées notamment sur la façon de créer un historique pour les variables dont on ne disposait que de la dernière valeur. D'une manière plus générale, pour que ce genre de méthodes puissent être utilisées, il est nécessaire d'avoir une base de données de très grande qualité. De plus, il serait intéressant, et cela sera sans doute possible avec le développement des nouvelles technologies, que les compagnies d'assurance effectuent une sauvegarde de leurs bases de données à intervalle de temps régulier. Cela leur permettrait d'avoir une vue de l'évolution de leur portefeuille à différentes étapes, de pouvoir justifier plus facilement différents calculs effectués dans le passé auprès des régulateurs et de faciliter le développement de l'utilisation d'algorithmes innovants qui pourraient leur être utile. Une autre limite de cette étude repose sur la modélisation, et notamment le choix de l'horizon de temps jusqu'à l'ultime. Cette hypothèse a eu un impact important sur la construction même de la base et soulève aussi d'autres problèmes.

En effet, en se projetant à cinq ans par exemple, cela implique d'avoir besoin d'une hypothèse sur le traitement de l'inflation sur les cinq années à venir. De plus, toutes les branches ne pourront pas avoir le même horizon de temps car les sinistres se développent différemment d'une branche à l'autre. Ainsi, le choix de l'horizon de temps à l'ultime est un choix important qui aura un impact directement sur la modélisation et également sur le retraitement des résultats.

Enfin, les résultats et autres commentaires de cette étude sont basés sur un unique jeu de données. Toutefois, chaque portefeuille d'assurance possède ses propres particularités. Il serait donc intéressant de tester ces approches sur d'autres portefeuilles de la GAV et ensuite sur d'autres types de garanties proposées par les assureurs.

Bibliographie

[1] Parodi P., *Triangle-free reserving : A non-traditional framework for estimating reserves and reserve uncertainty*, 2014, British Actuarial Journal, 19(1), 168-218.

[2] Robert Kroon, *Individual Reserving by Detailed Conditioning - A parametric approach*, BSc Acturial Sciences, non publié.

[3] Godecharle, Els and Antonio, Katrien, *Reserving by Conditioning on Markers of Individual Claims : A Case-Study Using Historical Simulation*, april 2015, North American Actuarial Journal, 2015, 19(4), 273-288.

[4] E. Brochu, V. M. Cora, et N. de Freitas, *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning*, eprint arXiv :1012.2599, arXiv.org, [cs], dec. 2010.

[5] Nando de Freitas, *Machine learning - Bayesian optimization and multi-armed bandits*,[vidéo en ligne]. Youtube, 12/02/2013 [consulté le 30 juillet 2017]. 1 vidéo, 1h et 20mn disponible à : https ://www.youtube.com/watch ?v=vz3D36VXefIindex=10list=PLE6Wd9FR–EdyJ5lbFl8UuGjecvVw66F6

[6] IASB, *IFRS 17 Insurance Contracts* ,may 2017, IFRS Standards.

[7] IASB, *IFRS 17 Insurance Contracts* ,may 2017, IFRS Standards Effects Analysis.

[8] IASB, *IFRS 17 Insurance Contracts* ,may 2017, IFRS Standards Project Summary.

[9] IASB, *IFRS 17 Insurance Contracts* ,may 2017, IFRS Standards Fact Sheet.

[10] IASB, *IFRS 17 Insurance Contracts* ,may 2017, IFRS Standards Feedback Statement.

[11] Darrel Scott, Andrea Pryde, *IFRS 17 Insurance Contracts* ,may 2017, IFRS Foundation.

[12] Stephen Cooper, Joan Brown, *Presenting insurance contract revenue*, 2012.

[13] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*, Second Edition, Springer, 2009, Springer Series in Statistics.

[14] Jerome Friedman, *Multivariate Adaptative Reggression Splines* , 2009, The Annals of Statistics, Vol 19.

[15] Jerome Friedman, *Estimating Functions of Mixed Ordinal and Categorical Variables Using Adaptive Splines*, Stanford university, Department of Statistics, Laboratory for computanional statistics, juin 1991, Technical report. no. 108.

[16] Ryan M. S., Nudd G. R., *The Viterbi algorithm. University of Warwick*, Department of Computer Science, Department of Computer Science research report, non publié.

[17] Sepp Hochreiter, Jürgen Schmidhuber, *Long Short-Term Memory*, Neural Computation 1735-1780, 1997.

[18] Felix A. Gers, Jürgen Schmidhuber, Fred Cummins, *Learning to Forget : Continual prediction with LSTM*, Neural Computation, oct. 2000.

[19] Tianqi Chen, Carlos Guestrin , *XGBoost : A Scalable Tree Boosting System*, eprint arXiv :1603.02754v3 [cs.LG], jun. 2016.

[20] Jerome H. Friedman, *Greedy Function Approximation : A Gradient Boosting Machine*, The Annals of Statistics, Vol. 29, No. 5 , pp. 1189-1232, oct. 2001.

[21] Yoav Freund, Robert E. Schapire, *A Short Introduction to Boosting* , The Annals of Statistics, Journal of Japanese Society for Artificial Intelligence, p771-780, sept. 1999.

[22] *Universal approximation theorem* , Wikipedia, jun. 2017.

[23] Rohan Kapur, Lenny Khazan, *Rohan & Lenny #3 : Recurrent Neural Networks LSTMs : The ultimate guide to machine learning's favorite child.*, A year of AI [en ligne],2017. Disponible sur : https ://ayearofai.com/rohan-lenny-3-recurrent-neural-networks-10300100899b

[24] Christopher Olah, *Understanding LSTM Networks*, colah's blog [en ligne],aug. 2015. Disponible sur : http ://colah.github.io/posts/2015-08-Understanding-LSTMs/

[25] Adrien Suru, *Cours : Principes de l'assurance dommage*, Université Paris Dauphine, M2 Actuariat - Département MIDO, 2016-2017.

[26] Clément Billoré, *Application de l'apprentissage automatique au provisionnement ligne à ligne en assurance non-vie*, Mémoire présenté devant le jury de l'EURIA en vue de l'obtention du Diplôme d'Actuaire EURIA et de l'admission à l'Institut des Actuaires, sept. 2016.

[27] Selma Jaziri, *Méthodes de provisionnement non-vie et risque de réserve à un an*, Mémoire présenté devant l'Institut de Science Financière et d'Assurances pour l'obtention du diplôme d'Actuaire de l'Université de Lyon, jul. 2011.

[28] Waes McKinney, *Python for data Analysis*, Second Edition, Springer, 2013, Agile Tools for Real-World Data.

Lexique

ANN : ANN est l'abréviation de réseau de neurones articiels (ou « vanille »). Il s'agit d'une structure algorithmique regroupant des neurones formels en différentes couches.

Année de développement : L'année de développement fait référence au délai en années depuis la survenance du sinistre.

Année de survenance : L'année de survenance correspond à l'année d'origine du sinistre.

Anti-sélection : Tendance d'un portefeuille à attirer ou à conserver des assurés avec des profils plus risqués que ne le souhaiterait l'assureur.

Apprentissage supervisé : Technique d'apprentissage automatique où l'on cherche à produire automatiquement des règles à partir d'une base de données d'apprentissage contenant des « exemples ».

Arbre de régression : Ensemble de règles de décisions, présentées sous la forme d'un arbre, permettant de réaliser une prédiction.

Assurance : Contrat par lequel l'assureur s'engage à indemniser l'assuré contre la survenance d'un évènement incertain qui lui serait défavorable, en contrepartie du paiement d'une prime ou d'une cotisation.

Assurance Non-vie : L'assurance non-vie regroupe les opérations qui n'ont pas pour objet la vie de l'assuré.

Couche (d'un réseau de neurones) : Ensemble de neurones formels n'ayant pas d'interactions entre eux mais avec tous les neurones des couches précédentes et suivantes lorsqu'elles existent

Échantillon : Ensemble d'éléments extraits d'une population étudiée.

GAV : La garantie contre les accidents de la vie (GAV) vise à protéger l'assuré, et éventuellement sa famille, des conséquences des accidents de la vie quotidienne.

IASB : IASB sont les initiales d'International Accounting Standards Board. Il s'agit de l'organisme chargé d'élaborer des normes comptables internationales.

IFRS : IFRS est l'abréviation d'International Financial Reporting Standards. Les normes IFRS sont les normes internationales d'informations financières destinées à standardiser la présentation des données comptables échangées au niveau international.

Neurone formel : Un neurone formel est une représentation mathématique et informatique d'un neurone biologique. Le neurone formel possède généralement plusieurs entrées et une sortie qui correspondent respectivement aux dendrites et au cône d'émergence du neurone biologique.

Rééchantillonner : Créer un nouvel échantillon à partir d'un déjà existant.

RNN : RNN est l'abréviation de réseaux de neurones récurrents. Il s'agit d'un réseau de neurones formels possédant une mémoire.

Sinistre : Évènement entraînant des dommages ou des pertes susceptibles d'être indemnisés.

Sinistre fermé : Sinistre pour lequel l'assureur et l'assureur ont trouvé un accord pour le dédommagement. Il n'y a donc plus de flux financiers associés.

Sinistre non garanti : Sinistre pour lequel les garanties prévues dans le contrat d'assurance ne sont pas applicables.

Sinistre ouvert : Sinistre déclaré par l'assuré et pour lequel l'assureur effectue un suivi.

Sinistre réouvert : Sinistre qui a été fermé avant d'être de nouveau ouvert.

Annexe A

Annexes de la partie estimation des provisions à l'aide de méthodes d'apprentissage automatique

A.1 Moments des distributions de la famille exponentielle

Pour rappel on a :

$$f(y|\theta,\phi) = a(y,\phi)exp(\frac{y\theta - k(\theta)}{\phi})$$

A.1.1 Moment d'ordre 1

On souhaite montrer que $E[Y] = k'(\theta)$.

$$\frac{\partial}{\partial\theta}\ln(f(y|\theta,\phi)) = \frac{\partial}{\partial\theta}(a(y,\phi) + \frac{y\theta - k(\theta)}{\phi}) \tag{A.1}$$

$$= \frac{y - k'(\theta)}{\phi} \tag{A.2}$$

Comme :

$$\int_{y\in\mathbb{R}} f(y|\theta,\phi)\,\mathrm{d}y = 1 \tag{A.3}$$

Et :

$$\begin{aligned}\frac{\partial}{\partial\theta}\int_{y\in\mathbb{R}} f(y|\theta,\phi)\,\mathrm{d}y &= \int_{y\in\mathbb{R}} \frac{\partial}{\partial\theta} f(y|\theta,\phi)\,\mathrm{d}y \\ &= \int_{y\in\mathbb{R}} (\frac{\partial}{\partial\theta}\ln f(y|\theta,\phi)) f(y|\theta,\phi)\,\mathrm{d}y \\ &= E[\frac{\partial}{\partial\theta}\ln f(Y|\theta,\phi)] \\ &= 0\end{aligned} \tag{A.4}$$

Donc :

$$
\begin{aligned}
E[\frac{\partial}{\partial\theta}\ln f(Y|\theta,\phi)] &= \frac{E[Y]-k'(\theta)}{\phi} \\
&= 0
\end{aligned}
\tag{A.5}
$$

D'où $E[Y] = k'(\theta)$.

A.1.2 Moment d'ordre 2

On veut maintenant montrer que $V(Y) = \phi k^{"}(\theta)$. Grâce à l'équation $(A.4)$ on a :

$$
\begin{aligned}
V[\frac{\partial}{\partial\theta}\ln f(Y|\theta,\phi)] &= E[(\frac{\partial}{\partial\theta}\ln f(Y|\theta,\phi))^2] \\
&= E[(\frac{Y-k'(\theta)}{\phi})^2] \\
&= \frac{V(Y)}{\phi^2}
\end{aligned}
\tag{A.6}
$$

De plus :

$$
\begin{aligned}
E[(\frac{\partial}{\partial\theta}\ln f(Y|\theta,\phi))^2] &= \int_{y\in\mathbb{R}}(\frac{\partial}{\partial\theta}\ln f(y|\theta,\phi))^2 f(y|\theta,\phi)\,\mathrm{d}y \\
&= \int_{y\in\mathbb{R}}\frac{\partial}{\partial\theta}\ln f(y|\theta,\phi)\frac{\partial}{\partial\theta}f(y|\theta,\phi)\,\mathrm{d}y \\
&= E[-\frac{\partial^2}{\partial\theta^2}\ln f(y,\theta,\phi)] \\
&= \frac{k^{"}(\theta)}{\phi}
\end{aligned}
\tag{A.7}
$$

Et donc :

$$
\begin{aligned}
V[\frac{\partial}{\partial\theta}\ln f(Y|\theta,\phi)] &= E[-\frac{\partial^2}{\partial\theta^2}\ln f(y,\theta,\phi)] \\
&= \frac{k^{"}(\theta)}{\phi}
\end{aligned}
\tag{A.8}
$$

En utilisant $(A.6)$ et $(A.8)$ on obtient le résultat souhaité.

A.2 Expression de k

On a $\mu = E[Y] = k'(\theta)$. On a aussi $V(Y) = \phi k^{"}(\theta)$. Or, d'après les notations de la distribution de Tweedie on a aussi : $v(Y) = \phi\mu^p$ et donc $k^{"}(\theta) = \mu^p$.

De là on en déduit que $k^{"}(\theta) = k'(\theta)^p$.

La primitive serait donc :

$$k(\theta) = \frac{c_1 - (-(p-1)(c_2+\theta)^{\frac{2-p}{1-p}}}{p-2} \tag{A.9}$$

où c_1 et c_2 sont deux constantes que l'on peut considérer arbitrairement comme étant nulles. On a alors :

$$k(\theta) = \frac{1-p}{2-p}\theta^{\frac{2-p}{1-p}} \tag{A.10}$$

En posant :

$$\theta = \frac{1}{1-p}\mu^{1-p} \tag{A.11}$$

On obtient :

$$k(\theta) = \frac{1}{2-p}\mu^{2-p} \tag{A.12}$$

A.3 Log-vraisemblance de la distribution Tweedie

Pour ϕ constant on a :

$$\begin{aligned} l(\mu) &= \log(\prod_{i=1}^{n} f(y_i|\phi,p,\mu) \\ &= \sum_{i=1}^{n} \log(f(y_i|\phi,p,\mu)) \\ &= \sum_{i=1}^{n} \log(a(y_i,\phi)\exp(\frac{1}{\phi}(\frac{y_i\mu^{1-p}}{1-p} - \frac{\mu^{2-p}}{2-p}))) \\ &= \sum_{i=1}^{n} \log(a(y_i,\phi)) + \sum_{i=1}^{n} \frac{1}{\phi}(\frac{y_i\mu^{1-p}}{1-p} - \frac{\mu^{2-p}}{2-p}) \end{aligned} \tag{A.13}$$