Analyse statistique de l'évolution des sinistres graves pour une garantie risque corporel

Isaac Cohen Sabban

Ecole doctorale de sciences mathématiques de Paris centre

Laboratoire de Probabilités, Statistique et Modélisation, LPSM, Sorbonne Université

THÈSE DE DOCTORAT

Discipline : Mathématique

Spécialité doctorale : Statistique

présentée et soutenue publiquement par

Isaac Cohen Sabban

Analyse statistique de l'évolution des sinistres graves pour une garantie risque corporel

---

Directeur de thèse : **Olivier Lopez**

le 1er Juillet 2022

Jury

M. Olivier Lopez,	Professeur, Sorbonne Université	Directeur de thèse
M. Bertrand Michel,	Professeur, Ecole Centrale Nantes	Rapporteur
M. Alexandre Brouste,	Professeur, Université Le Mans	Rapporteur
M. Idris Kharroubi,	Professeur, Sorbonne Université	Jury
M. Yann Mercuzot,	Responsable Actuariat, Pacifica	Jury

Analyse statistique de l'évolution des sinistres graves pour une garantie risque corporel.

Analyse statistique de l'évolution des sinistres graves pour une garantie risque corporel.

Laboratoire de Probabilités, Statistique et Modélisation (LPSM)
Sorbonne Université
Tours 15-25,2ème étage
4 place Jussieu
75252 Paris Cedex 05

PACIFICA
8 10 Boulevard de Vaugirard
75015 Paris

À mon père, ma mère et mes sœurs

Remerciements

Mes premiers mots vont naturellement à mon directeur de thèse, Olivier Lopez. Après mon inscription en première année de thèse, tu m'as dit « La thèse, c'est trois longues années durant lesquelles tu te sentiras seul ». Pour une fois, je peux te dire que tu as tort. Je tiens à te remercier pour tout ton soutien durant cette thèse qui fut longue et difficile à conclure. Tu as été une source de motivation et un véritable puits de connaissance. Merci pour tes conseils et pour ton encadrement qui ont facilité les travaux et l'écriture de cette thèse tout au long du parcours.
Merci également à Maud Thomas, Maitre de conférence à Sorbonne Université, pour le temps que tu m'as consacré à partager ton savoir sur la Théorie des Valeurs Extrêmes.

Je remercie également Bertrand Michel et Alexandre Brouste d'avoir accepté de rapporter ma thèse. Merci pour vos commentaires positifs. Je tiens également à te remercier Bertrand en tant que professeur, c'est suite à nos échanges et à tes cours que j'ai décidé de prendre une carrière plus orientée vers les statistiques et la recherche.

Je voudrais également remercier la direction de PACIFICA, Lionel Feraud, Directeur du marché des particuliers, et Yann Mercuzot, responsable du secteur Actuariat produits et réassurance, pour leurs accueils au sein de la DPART.
Merci également à Audrey Mahuzier et Marie Tien, pour le temps, l'énergie et la confiance que vous m'avez consacrés. Ce fut un vrai plaisir de travailler à vos côtés.
J'en profite pour remercier tous mes collègues de Pacifica avec qui j'ai eu le plaisir de travailler et d'échanger. Merci aux membres du CEDS, Charles et Clément, pour vos échanges sur la data science.

Je tiens à remercier Idris Kharroubi d'avoir accepté d'être examinateur de mon travail de thèse.

Je remercie les doctorants du LPSM pour tous les bons moments que nous avons partagés. Les conférences, les séminaires, les multiples discussions que nous avons eues sur divers sujets. J'espère pourvoir encore échanger avec vous et vous souhaite bonne chance dans vos recherches.
Merci à toute l'équipe administrative du laboratoire et de l'école doctorale.

Ce manuscrit a été amélioré par les remarques de plusieurs proches que je tiens à remercier. Sans votre aide, ce document ne serait pas ce qu'il est aujourd'hui. Merci à vous pour vos conseils et vos échanges constructifs.

Cette thèse marque la fin de mes études universitaires. Je suis rentré à l'Université Pierre et Marie Curie Paris 6 en septembre 2010, j'en ressors aujourd'hui avec certes des diplômes mais aussi avec de très bons amis qui m'ont soutenu et accompagné durant cette thèse, et je vous en remercie.

Avant de conclure ces remerciements, je tiens à remercier celle qui m'a accompagné durant cette thèse, celle qui m'a tellement entendu en parler et vu m'entrainer qu'elle pourrait presque la soutenir à ma place. Merci à toi.

Et pour finir, je remercie mes parents, Inès et Mosché Cohen Sabban, ainsi que mes sœurs, Léa, Dina et Karen, pour leur soutien et leur accompagnement depuis le début. Un jour, maman, tu m'as demandé de ne pas faire médecine car tu pensais être trop vieille à la fin de mes études. Désolé...

Table des matières

Remerciements **11**

1 Introduction **17**

1.1 L'importance du Reserving en assurance 19

1.1.1 Le contexte Solvabilité II 19

1.1.2 Le rôle des régulateurs 21

1.1.3 Le temps en assurance 22

1.1.4 L'importance du provisionnement 23

1.1.5 Les provisions techniques 24

1.1.6 Les sinistres extrêmes : une limite de la mutualisation du risque 25

1.2 Les modèles de reserving en assurance 26

1.2.1 Les modèles agrégés 27

1.2.2 Conclusion sur les méthodes agrégées 30

1.2.3 Les modèles de Micro Level reserving 31

1.3 La théorie des valeurs extrêmes (TVE) 32

1.3.1 Notations 33

1.3.2 Loi du Maximum 34

1.3.3 La loi des excès et la Loi de Pareto Généralisée 39

1.3.4 Estimation du seuil 40

1.4 Les modèles de durée 45

1.4.1 Quelques notions importantes 45

1.4.2 L'estimateur de Kaplan-Meier 46

1.4.3 Les poids IPCW 47

1.4.4 Le modèle de Cox 48

1.5 Les Modèles Linéaires Généralisés 48

1.5.1 Hypothèses du modèle linéaire généralisé 49

1.5.2 Le modèle logistique 50

1.5.3 Loi Gamma 51

1.5.4 Régularisation . . . 51
1.6 Les modèles de Machine Learning . . . 53
1.6.1 L'algorithme CART . . . 54
1.6.2 Les méthodes ensemblistes . . . 54
1.7 Introduction aux Réseaux de neurones . . . 57
1.7.1 Introduction au Deep Feedforward Networks . . . 58
1.7.2 CNN : Convolutional Neural Networks . . . 62
1.7.3 RNN : Recurrent Neural Networks . . . 65
1.8 Travailler avec des données déséquilibrées . . . 70
1.8.1 Undersampling . . . 70
1.8.2 Oversampling . . . 71
1.8.3 Recommandation sur l'Oversampling et l'Undersampling . . . 73
1.8.4 Stratégies Algorithmiques . . . 73
1.8.5 Le choix de la métrique . . . 74
1.9 Organisation de la thèse . . . 75
1.9.1 Chapitre 2 . . . 75
1.9.2 Chapitre 3 . . . 75
1.9.3 Chapitre 4 . . . 75
1.9.4 Chapitre 5 . . . 76

2 Présentation des données et premiers traitements 77
2.1 Les données . . . 77
2.1.1 Données Open Source . . . 80
2.2 Traitement des données structurées . . . 81
2.2.1 Traitement des données quantitatives . . . 82
2.2.2 Traitement des données qualitatives . . . 83
2.3 Traitement du texte . . . 83
2.3.1 Le nettoyage du texte . . . 84
2.3.2 Label encoder et One-hot encoding . . . 84
2.3.3 Word Embedding . . . 85
2.4 Théorie des valeurs extrêmes et détermination du seuil des sinistres graves . . . 86
2.4.1 Estimation du seuil . . . 94
2.4.2 Ajustement de la GPD . . . 97
2.4.3 Mise en as-if . . . 98
2.4.4 Gestion des sinistres en cours . . . 101

3 Automatic analysis of insurance reports through deep neural networks to identify severe claims. 107

3.1 Introduction . . . 107
3.2 Framework and neural network predictors . . . 108
3.2.1 The censoring framework . . . 109
3.2.2 Neural network under censoring . . . 111
3.2.3 Bagging for imbalanced data . . . 112
3.2.4 Summary of our methodology . . . 115
3.3 Embedding methodology . . . 115
3.3.1 One-hot encoding . . . 116
3.3.2 Word embedding . . . 116
3.4 Network architectures for text data . . . 118
3.4.1 Convolutional Neural Network . . . 118
3.4.2 Recurrent Neural Networks and Long Short Term Memory (LSTM) networks 119
3.5 Real data analysis . . . 121
3.5.1 Description of the database . . . 121
3.5.2 Hyperparameters of the networks and type of embedding . . . 123
3.5.3 Performance indicators . . . 125
3.5.4 Results . . . 125
3.5.5 Alternative methods for text analysis of imbalanced data . . . 129
3.6 Conclusion . . . 129

4 Prediction of the outcome of insurance claims with deep neural networks 131

4.1 Introduction . . . 131
4.2 Framework and general methodology . . . 132
4.2.1 Data . . . 132
4.2.2 A two steps procedure . . . 133
4.3 Structure of the prediction models . . . 134
4.3.1 Decomposition of the variable A into two modalities . . . 134
4.3.2 Feature extraction and computation of $\widehat{a}$ . . . 135
4.3.3 Final predictor . . . 135
4.3.4 Prediction of T . . . 136
4.4 Neural networks and Long Short Term Memory . . . 137
4.4.1 Neural networks terminology and notations . . . 137
4.4.2 Embedding layer . . . 138
4.4.3 Long Short Term Memory . . . 139
4.4.4 Extraction of features from the networks . . . 140

4.5 Real data application . . . 141
4.5.1 Description of the database . . . 141
4.5.2 Tail of the distribution of the loss . . . 142
4.5.3 Prediction models . . . 143
4.5.4 Classification of claims . . . 144
4.5.5 Claim amounts and reserving . . . 146
4.6 Conclusion . . . 148

5 Conclusion et perspectives **149**

A Chapitre 1 **152**
A.1 Descente de Gradient : Méthode de Newton-Raphson . . . 152
A.2 Méthode de Newton-Raphson . . . 152
A.2.1 Interprétation Géométrique . . . 153
A.2.2 Critère d'arrêt . . . 153
A.3 Évaluation d'un modèle de Régression . . . 153
A.3.1 GLM . . . 153
A.3.2 Métrique utilisé pour les méthodes de Machine Learning . . . 154
A.3.3 MSE . . . 155
A.4 Les étapes du LSTM . . . 156

B Annexe Chapitre 2 **157**
B.1 Graphiques complémentaires pour la TVE . . . 157
B.1.1 2002-2006 . . . 157
B.1.2 2007-2011 . . . 158
B.1.3 2012-2016 . . . 159

C Chapitre 3 **161**
C.0.1 Typical choices of activation functions for neural networks . . . 161
C.0.2 Additional type of layers in a CNN . . . 161
C.0.3 Hyperparameters . . . 162

D Chapitre 4 **164**
D.0.1 Duplication algorithm to correct censoring . . . 164
D.0.2 Hyperparameters . . . 165

Table des figures **169**

Liste des tableaux **171**

Chapitre 1

Introduction

Chaque individu est confronté à divers risques au cours de sa vie contre lesquels il souhaite se prémunir. L'assurance est l'opération par laquelle cet individu, l'assuré, transfère le risque qu'il encourt à l'assureur, dès lors que ce dernier peut en évaluer le coût et l'additionner à d'autres risques semblables. En France, la principale garantie pour laquelle l'engagement de l'assureur est illimité est la responsabilité civile Automobile, celle-ci couvre les dommages causés par l'assuré à un tiers.

En France, la charge corporelle relative aux sinistres Automobile représente environ 1/3 de la charge totale associée à un exercice de survenance[1]. De plus, les sinistres corporels ont subi une profonde mutation depuis le début des années 2000 avec une indemnisation toujours plus favorable des victimes, ce qui a conduit les assureurs à intégrer une inflation significative dans leurs modèles de prime pure. Cette inflation a été transportée pour certains dans leurs tarifs et pour d'autres dans la baisse de la marge de prudence existant dans leurs réserves. Si les « petits sinistres corporels » sont relativement normés et leur analyse plus directe en raison de leur liquidation rapide, il est à ce jour beaucoup plus difficile de disposer d'une analyse sans biais et d'un avis pertinent et étayé sur les sinistres majeurs à liquidation longue.

Une des difficultés que rencontre le marché de l'assurance est que, à l'inverse des marchés « classiques », comme le marché de l'industrie par exemple, l'assureur vend à l'assuré des services sans que leur coût (qui peut être nul) ne soit connu à l'avance, puisque les éventuelles prestations ou indemnisations qu'il devra verser dépendent de phénomènes aléatoires non encore réalisés. Ce principe spécifique à l'activité d'assurance est appelé l'inversion du cycle de production et donne lieu à d'importantes conséquences économiques et réglementaires pour l'assureur. Tout d'abord, les primes perçues par l'assureur sont déterminées à partir d'une estimation prévisionnelle des coûts des risques, sur laquelle réside une incertitude, ce qui représente un risque financier pour l'assureur. La période d'indemnisation des assurés pouvant s'étendre sur plusieurs années, ces primes sont ensuite placées sur les marchés financiers afin d'être fructifiées, en respectant néanmoins l'encadrement fixé par le code des assurances permettant d'en garantir la sécurité. De plus, du fait de cet inconnu sur l'avenir, l'assureur doit constituer des provisions techniques suffisamment prudentes pour pouvoir faire face à ses engagements envers ses assurés et ainsi payer des sinistres futurs. Afin de protéger les assurés, un cadre réglementaire a été mis en place pour obliger les assureurs à respecter des règles comptables et de gestion prudentielle spécifiques. Les autorités en charge (l'ACAM, Autorité de Contrôle des Assurances et des Mutuelles) veillent donc à ce que l'assureur soit solvable, autrement dit qu'il constitue des provisions techniques et des marges de solvabilité nécessaires et suffisantes selon le règlement en vigueur, actuellement la réforme Solvabilité II. Cette sécurité représente un coût commercial non négligeable pour l'assureur.

Le provisionnement est donc un élément qui se retrouve au cœur de l'activité d'assurance et qui est

essentiel à la survie de ces sociétés. La complexité de la modélisation des sinistres corporels nécessite l'utilisation de techniques innovantes. Dans cette thèse, nous proposons de mettre en œuvre des méthodes d'apprentissage statistique peu utilisées dans ce contexte. Ainsi qu'il sera détaillé plus loin dans ce projet, les contraintes propres au contexte des risques corporels devront être intégrées aux méthodes d'apprentissage considérées (durée de développement des sinistres, valeurs extrêmes…). Ceci débouchera sur la conception de nouveaux modèles venant enrichir la littérature scientifique sur le sujet.

En effet, les méthodes utilisées pour analyser le risque corporel dossier par dossier reposent actuellement principalement sur l'utilisation du modèle linéaire généralisé (GLM), ou éventuellement sur des méthodes bayésiennes comme la théorie de la crédibilité (voir par exemple Nelder and Verrall [150] (1997). Bien que ces méthodes puissent donner une certaine satisfaction sur d'autres types de risques moins volatils, et au temps de développement généralement plus courts (comme le risque dommages matériels), elles se révèlent néanmoins moins performantes notamment pour trois raisons :

- Leur incapacité à capter facilement des évolutions temporelles
- Leur difficulté à gérer des sinistres dont les coûts et les profils associés sont particulièrement hétérogènes (modèles excessivement paramétriques), notamment des valeurs extrêmes.
- Leur incapacité à définir des profils de risque (et les primes pures puis tarifs associé(e)s) faiblement ou fortement exposés aux sinistres corporels graves. Notre base de sinistres modélisée sur 10 ans de survenance est composée à 70% de provisions, c'est-à-dire d'une charge qui sera soumise à différentes évolutions avant sa clôture. Il en réside un inconfort certain puisqu'en cas de trajectoires de liquidation hétérogènes entre profils, les primes pures estimées sont donc biaisées et conduisent à une sous-optimalité de nos méthodologies de tarification et *in fine* de nos tarifs.
- Leurs faiblesses lorsque la dimension est importante, c'est-à-dire quand le nombre de facteurs explicatifs est important (or la complexité du risque nécessite précisément de rassembler d'importantes masses d'informations si on souhaite expliquer correctement le risque).

Cette faible performance et les signaux parfois contradictoires que peuvent envoyer ces modèles incite donc souvent à utiliser des approches agrégées (de type triangles de liquidation) qui sont régulièrement pointées du doigt comme peu précises (voir Jin, Frees (2013) [106], Antonio, Plat (2014)[8]).

Avec l'émergence d'outils informatiques de plus en plus performants, le monde de l'assurance s'est vu impacté sur deux aspects. Le premier est le type de données collectées par les assureurs. En effet, les images d'un sinistre ou les rapports liés à un sinistre sont un nouveau type de donnée que possède l'assureur. L'utilisation de texte ou d'images peut se révéler utile dans la prédiction de la charge. Le deuxième touche à la partie modélisation. Alors qu'historiquement, on contraignait les variables à "coller" à une loi de probabilité (modèles paramétriques), on a pu assister à l'émergence de nouveaux modèles qui sont basés sur les données (modèles non paramétriques).

La finalité de ce travail de thèse est de montrer comment la richesse des données disponibles sur des sinistres d'assurance (en l'espèce, données provenant de l'assureur Pacifica), peut être mise à profit pour améliorer significativement la prédiction du montant final d'un sinistre (ou son issue, lorsqu'on s'intéresse notamment à la classification d'un sinistre suivant son niveau de gravité). Pour traiter des données d'un tel volume, dont certaines sont des données textuelles - peu usuelles en assurance - nous utilisons des techniques d'apprentissage statistique (notamment réseaux de neurones profonds tels que les Convolutional Neural Networks ou des réseaux Long Short Term Memory). Par rapport à des applications plus usuelles, plusieurs difficultés se présentent à nous :

— l'importance de l'écoulement temporel des sinistres : du fait de la forte dépendance entre le montant final d'un sinistre et le temps nécessaire à sa gestion, des problématiques classiques en analyse de survie, comme celle des données censurées, vont se poser à nous.
— la rareté des événements sur lesquels nous souhaitons nous focaliser : les sinistres graves, qui sont notre préoccupation principale, sont peu nombreux, mais peuvent peser extrêmement lourd sur les comptes de l'assureur. Ce faible nombre, au sein de l'historique, rend difficile l'ajustement de méthodes d'apprentissage statistique, et nous serons contraints d'adapter ces méthodes pour prendre en compte des échantillons (très fortement) déséquilibrés.
— le vocabulaire spécifique des données textuelles traitées en assurance qui va nécessiter une calibration spécifique des méthodes d'encodage du texte.

L'organisation de ce chapitre est la suivante. Dans la section 1.1, nous préciserons les enjeux du provisionnement (ou reserving) en assurance, les outils que nous développons par la suite s'inscrivant dans cette finalité. Nous présentons dans la section 1.2 un bref aperçu des modèles utilisés en actuariat dans le but d'effectuer ce provisionnement. Notre problématique concernant les sinistres graves, la section 1.3 présente quelques résultats de théorie des valeurs extrêmes dont nous aurons besoin afin d'analyser la queue de distribution du montant des sinistres. Une autre spécificité de notre approche est de prendre en compte l'écoulement temporel des sinistres, particulièrement important lorsque l'on s'intéresse à une branche d'assurance à développement long comme la responsabilité civile. La section 1.4 présente quelques éléments d'analyse de survie qui nous serons utiles pour étudier ces phénomènes temporels. Enfin, les sections 1.5 à 1.8 s'intéressent à la question de la régression, puisque notre but est d'analyser ou de prédire l'impact de caractéristiques d'un sinistre sur son évolution. La section 1.5 revient sur les modèles paramétriques classiques que sont les modèles linéaires généralisés. La section 1.6 présente quelques approches plus modernes du machine learning, qui fournissent une modélisation plus riche des phénomènes, avec une présentation plus approfondie des réseaux de neurones en section 1.7. Notamment, nous nous intéresserons à leur capacité à analyser des données textuelles, puisque l'un de nos buts est d'incorporer l'analyse de rapports d'expertise dans la prédiction de l'issue d'un sinistre. Enfin, la section 1.8 se focalise sur l'aspect déséquilibré des jeux de données que nous allons considérer.

1.1 L'importance du Reserving en assurance

Afin d'expliquer le cadre de la problématique industrielle que nous considérons dans cette thèse, nous présentons tout d'abord quelques éléments de contexte réglementaire (sections 1.1.1 et 1.1.2). Nous expliquerons ensuite les difficultés techniques posées par la question du provisionnement (section 1.1.3) et l'importance du provisionnement dans l'activité d'assurance (section 1.1.4 et 1.1.5), en particulier des sinistres graves (section 1.1.6).

1.1.1 Le contexte Solvabilité II

La gestion des risques en assurance, et notamment sur les sinistres dits corporels, a pris un rôle plus important suite à la mise en place de Solvabilité II. Solvabilité II [80] est une réforme réglementaire européenne du monde de l'assurance, qui vise à mieux adapter les fonds propres exigés des compagnies d'assurance et de réassurance avec les risques que celle-ci encourent dans leur activité (risques de souscription, crédit, risque opérationnel et risque de marché) Elle fait suite à la crise financière de juillet 2007. Les régulateurs se rendent comptent de leurs erreurs et certains instituts financiers couleront tel que Lehman Brothers, ceux restant vont devoir subir de nouvelles règles plus restrictives.

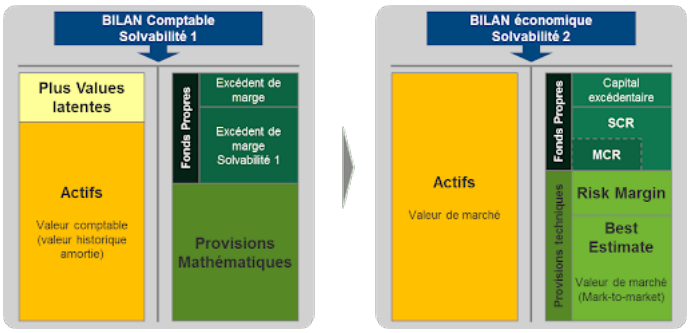


FIGURE 1.1 – Comparaison entre Solvabilité I et Solvabilité II

Un an après le début de cette crise, le cadre de Solvabilité II est dessiné. Là où Solvabilité I a échoué sur la marge de solvabilité qui était déterminé en fonction du pourcentage sur les primes et les sinistres, Solvabilité II intègre une quantification du risque plus complexe.

C'est un ensemble de règles édité par la Commission Européenne et adopté en 2009 par le Conseil de l'Europe et le Parlement Européen. Son but est de définir de nouvelles exigences de solvabilité des entreprises de façon harmonisée en Europe, et de les encourager à mieux connaître et évaluer les risques auxquels elles sont exposées. Cela passe notamment par la nécessité pour ces entreprises de mobiliser suffisamment de capital dans leurs fonds propres afin d'être solvable à horizon d'un an avec une probabilité 99,5 %. La solvabilité est la capacité d'une entreprise à faire face à ses engagements, c'est-à-dire générer suffisamment d'activité pour pouvoir rembourser ses dettes envers ses créanciers (dettes financières) et ses assurés (provisions techniques). Une partie du remboursement de ces engagements est assurée par les fonds propres, capital détenu par les propriétaires de l'entreprise. Ils constituent donc une sécurité dans les comptes, que la Commission Européenne voudrait renforcer en leur imposant des seuils minimum, le MCR et le SCR.

Solvabilité II repose sur trois piliers.

Le Pilier 1 détermine les exigences de capital qui s'appliquent aux assureurs depuis le 1er janvier 2016. Il repose sur quelques grands principes :

- Le calcul des provisions techniques en Best estimate. Cette méthode repose sur des modèles de projection définies à partir d'hypothèses crédibles (Nous les présenterons plus tard dans ce chapitre). Le but est de mesurer le risque d'occurrence de sinistres et de définir le montant juste de réserves nécessaires pour le couvrir. Le calcul de la meilleure estimation est fondé sur des informations actualisées et crédibles et des hypothèses réalistes et fait appel à des méthodes actuarielles et statistiques adéquates, applicables et pertinentes. Cela repose au préalable sur une méthode rigoureuse d'identification et quantification des risques impactant ces provisions, imposées via la formule standard ou la validation d'un modèle interne.
- L'élaboration d'un bilan prudentiel en valeur de marché.
- Des principes en matière d'allocation et d'exigibilité des actifs, au lieu de règles de répartition.
- Deux exigences de capital, le MCR (capital minimum requis) et le SCR (capital de solvabilité requis).
  - le MCR (Minimum Capital Requirement) représente le niveau minimum de fonds propres en dessous duquel l'intervention de l'autorité de contrôle sera automatique ;
  - le SCR (Solvency Capital Requirement) représente le capital cible nécessaire pour absorber le choc provoqué par un risque majeur (Un sinistre extrême par exemple...).

Le deuxième pilier a pour objectif de fixer des normes qualitatives de suivi des risques en interne aux sociétés et comment l'autorité de contrôle doit exercer ses pouvoirs de surveillance dans ce contexte.

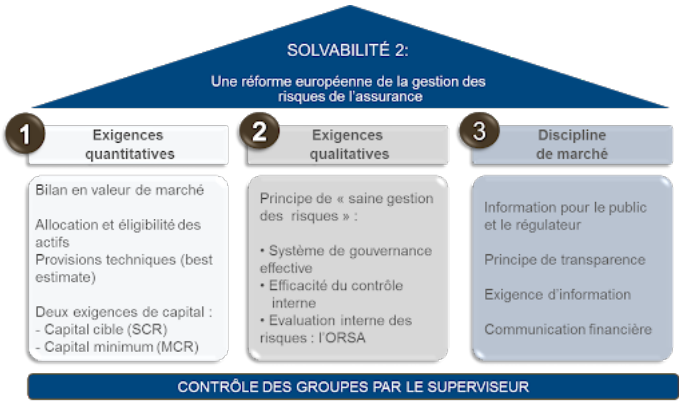


FIGURE 1.2 – Les trois piliers de Solvabilité II

L'identification des sociétés "les plus risquées" est un objectif et les autorités de contrôle auront en leur pouvoir la possibilité de réclamer à ces sociétés de détenir un capital plus élevé que le montant suggéré par le calcul du SCR (capital add-on) et/ou de réduire leur exposition aux risques.

Le troisième pilier a pour objectif de définir l'ensemble des informations détaillées auxquelles le public aura accès, d'une part, et auxquelles les autorités de contrôle pourront avoir accès pour exercer leur pouvoir de surveillance, d'autre part.

On parle souvent d'un pilier quantitatif, d'un pilier qualitatif, et d'un pilier de transparence et reporting. Afin d'honorer leurs engagements, les assureurs constituent des provisions mathématiques. Plusieurs contextes peuvent rendre difficiles l'estimation de ces provisions. Tout à d'abord l'aspect économique, les taux bas, une croissance faible, cela oblige les assureurs à naviguer à vue. Afin de s'assurer de la bonne santé financière des assureurs et du respect des règles de solvabilité, l'État a mis en place des régulateurs.

1.1.2 Le rôle des régulateurs

Le 9 mars 2010, l'État décide de créer l'ACP, qui en 2013 deviendra l'Autorité de contrôle prudentiel et de résolution (ACPR). Elle est issue de la fusion des autorités de contrôle des secteurs de la banque et de l'assurance et des autorités d'agrément. La création de cette autorité aux compétences élargies constitue une réponse aux nouveaux enjeux apparus lors de la crise financière de 2008 et dans son prolongement : la nouvelle autorité est explicitement chargée de veiller à la préservation de la stabilité du système financier, pour prévenir la survenue de nouvelles crises financières. L'ACPR s'assure que l'assureur reste en mesure d'honorer ses engagements pris envers ses clients. Pour cela, elle évalue les provisions techniques, la qualité de ses actifs et l'existence d'une marge de solvabilité permettant de faire face à d'éventuelles pertes futures. Si elle estime que l'assureur n'a pas une santé assez solide, elle peut lui enlever son autorisation d'exercer.

Les compagnies d'assurance jouent un rôle essentiel dans le développement social et économique d'un pays. En effet, elles contribuent d'une part à réduire la vulnérabilité des ménages en permettant l'accès à des services essentiels, dans le domaine de la santé notamment. D'autre part, un marché d'assurance solide accroît également la productivité des entre-prises, quelle que soit leur taille, en permettant aux entrepreneurs de saisir des opportunités plus risquées, mais dotées d'un potentiel de rendement supérieur. Par conséquent, la faillite d'un assureur aurait des impacts socio-économiques désastreux tant pour ses clients que pour l'État.

Afin d'éviter la faillite, les assureurs ont recours à une technique souvent pointée du doigt par les "bons assurés" qui estiment payer pour les mauvais, c'est la mutualisation du risque. Cette technique consiste à répartir le coût de la réalisation d'un sinistre entre les différents individus d'un groupe soumis potentiellement au même risque. Les primes reçues par l'assureur paient donc les sinistres de ses assurés. Le problème est que certains sinistres, tels que les pertes de biens dues à un incendie, sont facilement estimées et réglées rapidement. D'autres, comme l'estimation des dégâts corporels, peuvent durer plus longtemps.

1.1.3 Le temps en assurance

Le temps joue un rôle important dans la constitution des réserves de l'assureur. Prenons deux sinistres ayant la charge :

- Dans le premier cas, le sinistre se clôture la même année qu'il a été ouvert. L'assureur paie donc directement le sinistre.
- Dans le deuxième cas, le sinistre se clôture 10 ans après l'ouverture, l'assureur estime le coût du sinistre puis place une partie de la somme sur les marchés afin de faire fructifier cette somme jusqu'à atteindre l'année de paiement.

Dans notre exemple, l'assureur a eu deux stratégies différentes. Dans le premier cas, il a dû sortir directement de ses réserves le montant du sinistre. Dans le deuxième cas, il a dû sortir moins d'argent de ses réserves, car une partie a été placée. Il a donc dans ce cas constitué une réserve moins importante que dans le premier cas. Cet exemple nous montre donc l'importance du temps en assurance. Dans les applications que nous traiterons ultérieurement (issues de l'assurance Responsabilité Civile automobile), un sinistre sur deux a une durée de vie de six mois. Une telle statistique n'est pas totalement vraie. En effet, elle dépend beaucoup de la typologie du sinistre. Comme le montre le Tableau 1.1, plus le sinistre à un coût extrême (nous reviendrons plus tard sur cette notion) plus sa durée de vie est importante. De plus, par la faible représentation de ce type de sinistre dans notre base (2,5 %), ils n'ont qu'une influence faible sur la durée de vie médiane.

Il faut également noter que dans le cas d'un accident corporel, il faudra notamment attendre au préalable la stabilisation de l'état de santé de la victime, ou bien sa majorité dans le cas d'une victime mineure. À titre informatif, nous savons que le temps de consolidation (hors enfant) représente $2/3$ de la vie du sinistre.

Type de sinistre	Charge nulle	Charge normale	Charge extrême
Représentation dans la base	21,5 %	76 %	2,5 %
Moyenne	3	10	22
Variance	3	9	11
1er quartile	0	5	14
Médiane	2	8	21
3e quartile	5	15	30

TABLE 1.1 – Distribution de la durée de vie du sinistre en fonction de la charge. La corrélation entre la durée de vie et le coût d'un sinistre corporel semble indiquer que plus un sinistre dure plus celui-ci aura une charge importante.

La durée des règlements étant longue pour un sinistre corporel, cela oblige l'assureur à distinguer l'année de survenance du sinistre de l'année de règlement. C'est justement sur cette idée que repose la méthode de Chain-Ladder.

1.1.4 L'importance du provisionnement

Une société d'assurance fait face à de nombreux risques. Ces risques vont d'une importante sinistralité à une erreur humaine, en passant par le potentiel défaut d'un réassureur. Nous ne parlerons ici que du risque de provisionnement aussi appelé Reserving Risk.

Le risque de provisionnement est le risque lié à un mauvais provisionnement des prestations restantes à payer sur les sinistres. Concrètement, les sinistres sont payés sur plusieurs années, or les normes comptables sont telles que les sociétés font leurs comptes sur un exercice d'un an. On parle d'inversion du cycle de production.

L'assurance présente un cycle économique inversé, car on doit estimer la charge prévue en N avant celle-ci. En effet, l'assuré a acheté une prestation (versement d'une réparation) sans que le montant de cette réparation (qui peut être nul) ne soit connu, car la prestation dépend d'un phénomène aléatoire non encore réalisé. Les primes doivent permettre de faire face à un niveau normal de sinistres à payer, mais en pratique le risque à venir peut être illimité.

Bien que le calcul des primes repose sur une étude statistique, il peut être difficile de prévoir avec exactitude le nombre de sinistres et la typologie de ceux-ci, la crise sanitaire que nous traversons en est la preuve. C'est ce qu'on appelle l'inversion du cycle de production, car dans le monde de l'industrie par exemple, on connaît le coût de fabrication et on sait de quelle marge par produit on dispose.

Ce mode de fonctionnement impacte donc le bilan comptable d'une société d'assurance. L'assureur doit expliquer qu'il couvre bien les frais à venir avec les primes déjà reçues.

Les assureurs sont donc dans l'obligation d'établir des provisions sur les sinistres à payer ainsi que sur les sinistres dits tardifs. Ce sont des sinistres qui sont survenus à l'année N, mais qui seront potentiellement déclarés plus tard. Bien que ne représentant qu'un faible échantillon, l'assureur doit aussi prendre en charge des sinistres non connus et par conséquence leurs coûts.

Une compagnie d'assurance doit garantir sa solvabilité en intervenant à plusieurs niveaux. Tout d'abord, elle doit tarifer les produits proposés de manière prudente. Or le marché de l'assurance est un marché concurrentiel. Depuis la loi Hamon (2015) [95], l'assuré peut facilement résilier son contrat auto à la date de son choix, passé un an de contrat. Pour changer de couverture, il suffit de souscrire directement un contrat chez un nouvel assureur. Celui-ci prend alors en charge les démarches de résiliation dans un délai d'un mois et veille à la continuité de la couverture de l'assuré entre les deux contrats.

En ajoutant à cette situation l'apparition des comparateurs d'assurances et la vente par internet qui s'est considérablement développée, on obtient une concurrence de plus en plus importante. Dans ce contexte, les assureurs doivent présenter le prix le plus juste. Pour cela, il est nécessaire de maîtriser les coûts des produits qu'ils proposent et notamment les coûts engendrés par les sinistres. La prime commerciale que paie les assurés, est composée de la prime pure à laquelle s'ajoute des chargements commerciaux (frais de production, frais de gestion ...) et un chargement fiscal. La prime pure est donc un élément essentiel de la tarification d'un produit puisqu'elle représente le coût du risque de l'assuré. De tels effets impliquent donc un changement et un renouvellement du portefeuille de façon plus rapide. De plus, cela empêche à l'assureur d'avoir une vision à long terme, il ne peut donc pas compter sur ses primes futures. Il fonctionne donc en capitalisation, c'est-à-dire que les primes encaissées à l'année N doivent couvrir le coût des sinistres de l'année N, d'où l'importance de mutualiser les risques.

Comme dit plus haut, La durée des règlements pouvant être longue, on ne paie pas forcément les sinistres survenus à l'année N la même année, il peut arriver qu'on les paie à $N+1$,$N+2$ etc voir

$N+10$ pour les sinistres les plus extrêmes. L'assureur doit donc faire des réserves ou des provisions techniques. Ces provisions seront utilisées une fois le montant final du sinistre connu. Afin d'honorer ses engagements, la compagnie d'assurance doit donc être en mesure de régler les sinistres en année N, même si l'assuré résilie en $N+1$.

De par leurs spécificités, certains sinistres qui sont indemnisés rapidement, on parle de branche à déroulement court, et d'autres, comme le dommage aux personnes où des complications peuvent apparaître des années après le sinistre, sont dites à déroulement long.

Nous reviendrons ultérieurement sur les aspects méthodologiques du calcul de ces provisions (voir section 1.2). Néanmoins, précisons d'ores et déjà que parmi ces méthodes de calcul, les plus classiquement répandues découlent de la technique dite de Chain Ladder [11]. À travers la première description sommaire de cette technique que nous souhaitons effectuer, nous voulons souligner dès maintenant quelques points structurels qui montrent à la fois la difficulté du modèle de provisionnement et les insuffisances des méthodes standards, les travaux de cette présente thèse visant précisément à réduire l'impact de celles-ci.

La méthode Chain Ladder utilise les paiements cumulés notés $C_{i,j}$ pour les sinistres survenus l'année i (année d'origine) après j années (année de développement). L'estimation des paiements cumulés $\widehat{C_i}$ se fait en utilisant un facteur de développement, noté f_i, définit par la relation $f_i = \frac{C_{i,j+1}}{C_{i,j}}$

La méthode de Chain-Ladder repose sur l'hypothèse que les facteurs de développements sont indépendants de l'année d'origine.

Dans le cas de l'estimation de ce facteur, on a la relation suivante $\widehat{f_j} = \frac{\sum_{i=0}^{n-j-1} C_{i,j+1}}{\sum_{i=0}^{n-j-1} C_{i,j}}$ et on obtient donc l'estimateur des paiements cumulés par le calcul suivant :

$$\widehat{C_i} = C_{i,n-i} * \widehat{f}_{n-i} * ... * \widehat{f}_{n-1}$$

Cette méthode est simple à mettre en place et assez intuitive. En revanche, elle présente l'inconvénient d'être déterministe. En effet, comme les montants cumulés des années précédentes sont connus et fixes, cela implique que les facteurs de d'enveloppements le sont aussi. L'estimation du provisionnement dépendant de ces facteurs, elle le devient également. Il existe d'autres méthodes de provisionnement plus élaborées prenant en compte ces variations, comme les méthodes de provisionnement individuel ou la méthode de Mack, variante stochastique des Chain-Ladder que nous présentons un peu plus tard dans cette thèse.

Par ailleurs, ces méthodes présentent une vision agrégée des montants des sinistres, qui permet difficilement d'exploiter les informations précises dont l'on dispose sur les circonstances de ceux-ci. Or un gain important de précision peut être espéré à partir d'une prise en compte de cette masse d'informations.

1.1.5 Les provisions techniques

Les provisions techniques représentent les engagements de l'assureur vis-à-vis de ses clients. Il existe plusieurs types de provisions :

Provision pour prime non acquise : Elle représente la part des primes émises et des primes restant à émettre se rapportant à la période comprise entre la date d'inventaire et la date de la prochaine échéance de prime ou, à défaut, du terme du contrat.

Provision pour risque en cours : Elle vise à se couvrir contre un risque de modèle

Provision pour sinistre à payer : Elle vise à couvrir le coût des sinistres survenus, mais pas encore payer. On distingue ici deux types de sinistres

- Ceux qui se sont produits et ont été déclarés à l'assureur. C'est ce qu'on appelle un RBNS, Reported But Not Settled, c'est un sinistre connu, mais non payés. C'est ceux-là dont on va chercher à améliorer le provisionnement.
- Ceux qui se sont produits et n'ont pas été déclarés à l'assureur. C'est ce qu'on appelle un IBNR, Incurred But Not Reported, c'est un sinistre inconnu.

On peut le voir, il y a une grande partie d'aléatoire dans les engagements de l'assureur, de par les montants, mais aussi de par la durée de ses engagements. Ces provisions sont réévaluées durant toute la vie du sinistre à la hausse (mali) ou à la baisse (boni).

1.1.6 Les sinistres extrêmes : une limite de la mutualisation du risque

La mutualisation des cotisations et des risques est l'un des piliers de l'assurance. Pour les assureurs, il s'agit d'équilibrer les clients ayant un fort risque avec ceux ayant un faible risque. Cela permet de diminuer les contraintes individuelles sur les assurés, mais peut provoquer un phénomène d'aléa moral (l'aléa moral intervient lorsque l'assuré modifie son comportement en fonction de la couverture assurantielle, la conscience d'une garantie en cas de sinistre engendrant un comportement moins prudent). Cette mutualisation s'avère opérationnelle puisque les risques sont dispersés.

Pour que la mutualisation fonctionne, l'assureur constitue des classes de risques. Ces classes sont basées sur des caractéristiques de l'individu, comme son âge, son type de véhicule ou bien ses antécédents en termes de sinistralité. Cette homogénéité s'entend donc relativement à un ensemble de variables explicatives convenablement sélectionnées, en général à l'aide de modèles linéaires généralisés. Cette homogénéité peut se mesurer par des indicateurs de risque, à savoir la fréquence et le coût moyen des sinistres qui permettent de déterminer la prime pure (produit de ces deux indicateurs).

Avant d'aller plus loin, nous souhaitons rappeler quelques définitions et notions d'assurance.

Définition 1. ***Cotisations acquises de l'exercice N :*** *Total des cotisations annuelles associées à chaque contrat calculées au prorata par rapport au nombre de jours d'assurance comptabilisés sur l'exercice N / 365 (ou 366 en année bissextile).*

Définition 2. ***Années assurances de l'exercice N :*** *Total des contrats assurés calculés au prorata unitaire par le ratio entre le nombre de jours d'assurance comptabilisés sur l'exercice N divisé par 365 ou 366.*

Définition 3. ***Charge :*** *Règlement du sinistre si celui-ci est clôturé ou évaluation de celui-ci s'il est en cours.*

Définition 4. ***Nombre de sinistres :*** *Total des sinistres survenus durant la période analysée (du 01/01/N au 31/12/N), hors sinistres annulés sauf si un règlement est présent au dossier. Au niveau garanti, il correspond au nombre de garanties associées aux sinistres survenus.*

Définition 5. ***Coût Moyen :*** *Coût total des sinistres présents dans le portefeuille d'un assureur divisé par le nombre de sinistres il est souvent noté CM et défini par la formule suivante :* $C.M = \frac{Charge}{Nombre\ de\ sinistres}$

Définition 6. ***Fréquence :*** *En assurance, cela désigne un nombre d'évènements qui se sont produits sur une certaine période rapportée à l'exposition du portefeuille sur cette période. Elle est définie par la formule suivante :*

$$Freq = \frac{Nombre\ de\ sinistres}{Nombre\ d'annees\ d'assurance}$$

Autrement dit, la fréquence équivaut à la probabilité d'un individu d'avoir un sinistre.

Définition 7. ***La Prime Pure :*** *La prime pure (notée PP) est le coût du risque, il s'agit du montant dont doit disposer l'assureur pour dédommager (en moyenne) les assurés suite aux sinistres survenus, sans excédent, ni déficit. La totalité des primes pures relatives au portefeuille doit pouvoir être couverte par l'assureur.*

La Prime Pure est calculée en prenant en considération différents éléments : probabilité de survenance ou fréquence des sinistres, étendue des préjudices, somme assurée, etc.

Définition 8. *La Prime Pure est définie par la formule suivante :*

$$PP = CM * Freq$$

ou d'après les définitions précédentes :

$$PP = \frac{Charge}{Nombre\ d'annees\ d'assurance}$$

La fréquence, le coût moyen et donc la prime pure, sont des indicateurs de risque du portefeuille. Une stabilité temporelle de ces indicateurs est donc nécessaire pour avoir une bonne adéquation entre la sinistralité et la tarification.

La présence d'un sinistre rare, c'est-à-dire un sinistre dont la charge ne correspondrait pas au risque attendu lors de la tarification, viendrait perturber cette hypothèse d'homogénéité des classes et donc la stabilité des indicateurs. La perte de la stabilité de ces indicateurs implique un risque de provisionnement. En général, face à de tels événements, les assureurs répartissent la charge sur l'ensemble du portefeuille. Toutefois, la question se pose de déterminer ce qu'est un sinistre grave et c'est principalement le rôle de la théorie des valeurs extrêmes qui est de déterminer ce qu'est un sinistre grave et comment le modéliser.

1.2 Les modèles de reserving en assurance

En assurance non-vie, le provisionnement est calculé comme la somme des sinistres qui ne sont pas encore produits et des primes non payées. Les sinistres prédits ne sont pas pris en compte. Il existe deux catégories de sinistres :

- Soit ils se sont produits et ont été déclarés à l'assureur. Celui-ci les a donc estimés soit de façon agrégées, soit de façon individuelles
- Soit ils se sont produits et n'ont pas été déclarés à l'assureur. C'est ce qu'on appelle un IBNR, Incurred But Not Reported, c'est un sinistre inconnu. Pour ce type de sinistre, l'estimation se fait au niveau du portefeuille.

Les estimations des coûts sont faites soit de façon déterministe, soit en utilisant des méthodes stochastiques. Il est important de noter que les sinistres ne sont pas nécessairement payés immédiatement et en une fois. Plusieurs facteurs peuvent provoquer un paiement échelonné sur plusieurs

années, notamment pour une sinistre corporel. En effet, pour les sinistres les plus extrêmes, il peut arriver qu'il y ait une rente ou que le sinistre dure lui-même sur une période très longue, et des frais sont remboursés au fur et à mesure à l'assuré. Il existe pléthore de modèles, mais nous ne citerons que les plus connus, pour plus de modèle, nous invitions le lecteur à consulter par exemple les ouvrages de Denuit et Charpentier [56] aux éditions Economica.

1.2.1 Les modèles agrégés

Les méthodes agrégées reposent sur l'utilisation de triangles de développement. Ces triangles reflètent l'évolution des sinistres au cours du temps, que ce soit les paiements, les charges ou même le nombre de sinistres.

Nous fixons le cadre et les notations suivants :

1. $X_{i,j}$ le montant des paiements effectués pour les sinistres survenus l'année i (année d'origine) après j années (année de développement)
2. $C_{i,j}$ les paiements cumulés par les sinistres survenus l'année i (année d'origine) après j années (année de développement)

La relation entre le montant et les paiements cumulés est défini par $X_{i,j} = C_{i,j} - C_{i,j-1}$

On peut les représenter sous formes de triangles :

Année de survenance	Année de développement						Provisions
	0	1	...	j	...	n	
1	$C_{1,0}$	$C_{1,1}$	...	$C_{1,j}$	...	$C_{1,n}$	$R_1 = 0$
2	$C_{2,0}$	$C_{2,1}$	...	$C_{2,j}$	...	$\widehat{C_{2,n}}$	R_2
⋮	⋮	⋮		⋮		⋮	⋮
i	$C_{i,0}$	$C_{i,1}$		$\widehat{C_{i,j}}$	...	$\widehat{C_{i,n}}$	R_i
⋮	⋮	⋮		⋮		⋮	⋮
n	$C_{n,0}$	$\widehat{C_{n,1}}$	...	$\widehat{C_{n,j}}$	...	$\widehat{C_{n,n}}$	R_n

FIGURE 1.3 – Triangle des paiements cumulés

Les éléments du tableau représentent les montants cumulés des paiements effectués au cours de l'année calendaire "année de survenance i + année de développement j". La diagonale représente ce qui a été payé au cours d'une année calendaire. Les $\widehat{C_{i,j}}$ représentent l'estimation des paiements cumulés. On obtient les provisions de l'année i en appliquant la formule suivante : $R_i = \widehat{C_{i,n}} - C_{i,n-i}$
Nous décidons de présenter la méthode de Chain-Ladder ainsi que différentes variantes de cette méthode qui n'a cessé d'évoluer au fur et à mesure des années.

1.2.1.1 Méthode de Chain-Ladder

En 1938, E. Astesan [11] a formalisé la technique des triangles de développement, engendrant ainsi la méthode très populaire dite Chain-Ladder. C'est l'une des plus utilisées en assurance de par sa facilité à mettre en œuvre. Elle s'applique aux triangles de paiements cumulés. L'idée de cette méthode est que les sinistres survenus lors d'années de survenances différentes se développent de façon similaire.

Définition 9. *On appelle facteurs de développements, notés f_i, les rapports $\frac{C_{i,j+1}}{C_{i,j}}$*

La méthode de Chain-Ladder repose sur l'hypothèse que les facteurs de développement sont indépendants de l'année d'origine. Cela implique donc que la "vie" des sinistres se déroule de la même façon, peu importe l'année de survenance.

Pour vérifier cette hypothèse, on s'assure que pour toute année d'origine, et pour une année de développement donnée, le f_j soit approximativement constant.

Le calcul des provisions en utilisant la méthode de Chain-Ladder se fait en trois étapes :

1. Calcul des $\widehat{f_j} = \frac{\sum_{i=0}^{n-j-1} C_{i,j+1}}{\sum_{i=0}^{n-j-1} C_{i,j}}$
2. Calcul du $\widehat{C_i} = C_{i,n-i} * \widehat{f}_{n-i} * ... * \widehat{f}_{n-1}$
3. Calcul des provisions $R_i = \widehat{C_{i,n}} - C_{i,n-i}$

L'hypothèse d'indépendance utilisée dans la méthode Chain-Ladder est assez restrictive. En effet, les conditions suivantes doivent être réunies pour qu'elle soit valide [157] :

- Le passé doit être suffisamment régulier. C'est-à-dire pas de changement dans le mode de la gestion des sinistres ou dans l'inflation spécifique d'une branche. On a donc à faire à la fois à une limite économique, mais aussi industrielle.
- La branche doit être peu volatile : c'est donc une limite pour les sinistres graves qui eux sont très volatils de par leur définition.
- Les données du portefeuille doivent être nombreuses et fiables.

Ce qui fait donc surtout la force de cette méthode, c'est sa simplicité d'utilisation et son universalité pour les montants négatifs (un recours). Elle peut s'appliquer à des triangles de charges ou à des triangles de paiements comprenant des recours.

En revanche, à cause de ses hypothèses, elle présente le désavantage d'être un mauvais estimateur de la charge d'un sinistre en cas de sinistre extrême. De plus, la méthode de Chain-Ladder impose une hypothèse assez forte et contraignante sur les données. En effet, le nuage des points $(C_{i,j}, C_{i,j+1})_{i=0,...,n-j-1}$ doit former une droite passant par l'origine.

Pour alléger cette supposition contraignante et pour élargir le nombre des bases de données sur lesquelles on peut travailler, les méthodes auto-régressives supposent judicieusement que pour un j fixé, $C_{i,j+1}$ est une fonction affine de $C_{i,j}$ [119], c'est le cas de la méthode de London-Chair.

1.2.1.2 Méthode de London-Chair

La méthode de London-Chair est une variante de la méthode de Chain-Ladder. Introduite par Benjamin et Eagles [21] pour le calcul des réserves au Lloyd's en 1986. On suppose ici que $C_{i,j+1} = f_j C_{i,j} + a_j$. Cette hypothèse implique que les couples $(C_{i,j}, C_{i,j+1})_{i=0,...,n-j-1}$ ne doivent pas être obligatoirement alignés sur une droite passant par l'origine.

On peut donc estimer les coefficients en minimisant l'erreur quadratique :

$$(\widehat{a_i}, \widehat{f_j}) = \underset{a,f}{\arg\min} \sum_{i=0}^{n-j-1} (C_{i,j+1} - a - fC_{i,j})^2$$

On remarque que cela équivaut à estimer a et f par la méthode des moindres carrés. On pourrait donc utiliser les outils de validation du modèle linéaire classique si on avait suffisamment de données. Or travaillant sur les données du triangle, la puissance du test est très petite.

1.2.1.3 Méthode de London-Pivot

La méthode de London-Pivot est une version hybride de la méthode de Chain-Ladder et la méthode de London-Chain, elle est introduite par Straub [195].

On suppose ici que $C_{i,j+1} = f_j(C_{i,j} + a)$. Comme dans la méthode de London-Chain, on suppose donc que les couples $(C_{i,j}, C_{i,j+1})_{i=0,...,n-j-1}$ doivent être alignés sur une droite. À la différence que cette fois-ci, on décompose l'ordonnée à l'origine en $f_j a$.

On introduit donc pour l'estimation,

$$\Delta(a, f_0, ..., f_{n-1}) = \sum_{j=0}^{n--1} \sum_{i=0}^{n-j-1} ((C_{i,j+1} - f_j(C_{i,j} + a))^2)$$

On estime donc les coefficients en minimisant l'erreur quadratique.

$$(\widehat{a}, \widehat{f_0}, ..., \widehat{f_{n-1}}) = \underset{a,f_0,...,f_{n-1}}{\arg\min} \Delta(a, f_0, ..., f_{n-1})$$

1.2.1.4 La méthode de Mack

La méthode de Mack est la version stochastique de la méthode Chain-Ladder. Elle a été introduite par T.Mack [139] et permet une estimation des erreurs commises par la méthode Chain-Ladder [140]. C'est un modèle non paramétrique qui permet d'estimer une marge des erreurs commises par la méthode Chain-Ladder [140]. Il est aussi appelé distribution free Chain-Ladder, car aucune hypothèse de distribution n'est faite sur les composantes du triangle. Elle repose sur trois hypothèses :

1. $(C_{i1,j})_{i=0,...,n}$ et $(C_{i2,j})_{i=0,...,n}$ sont indépendants pour $i1 \neq i2$
2. $\exists f_i,\ \mathbb{E}[C_{i,j+1}|C_{i,j}, C_{i,j-1}, ..., C_{i,0}] = f_j C_{i,j}$
3. $\forall\, i, j\ Var[C_{i,j+1}|C_{i,j}, C_{i,j-1}, ..., C_{i,0}] = \sigma_j^2 C_{i,j}$

L'hypothèse 2 permet d'écrire le facteur de développement comme une v.a d'espérance conditionnelle : $\mathbb{E}[\frac{C_{i,j+1}}{C_{i,j}}|C_{i,j}, C_{i,j-1}, ..., C_{i,0}] = f_j$.

Par conséquence, l'hypothèse 3 devient $Var[\frac{C_{i,j+1}}{C_{i,j}}|C_{i,j}, C_{i,j-1}, ..., C_{i,0}] = \frac{\sigma_j^2}{C_{i,j}}$

En notant $\mathscr{F}$ la tribu engendrée par l'ensemble des $C_{i,j}$, pour $i+j \leq n$ (Autrement dit, les paiements déjà observés), on démontre facilement par récurrence que

$$\mathbb{E}[C_{i,n}|\mathscr{F}] = f_{n-1}...f_{n-i}C_{i,n-i}$$

On retrouve donc la même forme que l'estimateur de $\widehat{C_{i,n}}$ proposé par la méthode de Chain-Ladder standard.

Et donc les $\widehat{f}_j \frac{\sum_{i=0}^{n-j-1} C_{i,j+1}}{\sum_{i=0}^{n-j-1} C_{i,j}}$ dont des estimateurs sans biais et non corrélés de f_j.

En posant,$\mathscr{B} = \{C_{i,j} | j \leq k, i+j \leq n\}, 1 \leq k \leq n$, on peut réécrire l'hypothèse 2 de la façon suivante :

$$\mathbb{E}[C_{i,k+1}|\mathscr{B}] = f_k C_{i,k}$$

. En appliquant la formule de $\widehat{f}_k$, on a

$$\mathbb{E}[\widehat{f}_k|\mathscr{B}] = f_k$$

et donc

$$\mathbb{E}[\widehat{f}_k] = f_k$$

et on montre l'égalité suivante

$$j \leq k \mathbb{E}[\widehat{f}_j \widehat{f}_k] = \mathbb{E}[\widehat{f}_j]\mathbb{E}[\widehat{f}_k]$$

Les σ_j^2sont estimés par $\widehat{\sigma_j^2} = \frac{1}{n-j} \sum_{i=0}^{n-j-1} C_{i,j} (\frac{C_{i,j+1}}{C_{i,j}} - \widehat{f}_j)^2$ et $\widehat{C_{i,n}} = \widehat{f_{n-1}} ... \widehat{f_{n-i}} C_{i,n-i}$.

En utilisant la relation, $R_i - \widehat{R_i} = \widehat{C_{i,n}} - C_{i,n}$ on peut donc écrire, $MSE(\widehat{C_{i,n}}) = \mathbb{E}[(\widehat{C_{i,n}} - C_{i,n})^2 | \mathscr{F}] = MSE(\widehat{R_i})$.

Une estimation naturelle est donc

$$\widehat{MSE}(\widehat{C}_{i,n}) = \widehat{C}_{i,n} \sum_{j=n-i+1}^{n-1} \frac{\widehat{\sigma_i^2}}{\widehat{f_j^2}} \left(\frac{1}{\widehat{C}_{i,j}} + \frac{1}{\sum_{k=0}^{n-j-1} C_{k,j}} \right) = \widehat{MSE}(\widehat{R_i})$$

1.2.2 Conclusion sur les méthodes agrégées

L'utilisation des méthodes dérivées de Chain-Ladder est surtout due à sa facilité de mise en œuvre. Elles nécessitent peu de puissance de calcul et un niveau de granularité des données assez faible. Une implication de cet avantage est qu'elles sont bien établies et documentées. Utilisées par les actuaires depuis de nombreuses années, ces méthodes ont fait l'objet de nombreuses études. De plus, ces méthodes sont très pratiques pour la comptabilité et l'audit, et sont facilement compréhensibles.

Ce qui fait la force de ces méthodes, fait aussi ses faiblesses. En agrégeant les données, on engendre une perte d'information qui peut être importante et qui pourrait nous aider à améliorer nos prédictions. De plus, nous ne sommes pas capables avec ces méthodes de prédire au niveau individuel. Ainsi, nous avons une différence de traitement entre la tarification, qui permet d'obtenir le prix d'un contrat au niveau individuel et le provisionnement.

Ces méthodes peuvent être biaisées sous certaines conditions comme l'ont démontré Halliwell (2007) [91] et Schiegl (2015) [187]. Notamment, car elles ont tendance à surestimer les provisions et sont biaisées pour des structures temporelles non multiplicatives.

De plus, les hypothèses sur lesquelles elles reposent impliquent qu'il doit y avoir une stabilité dans le temps des coûts des sinistres. Premièrement, car elles ne prennent pas en compte l'évolution économique, mais aussi le changement de jurisprudence ou un changement de gestion de la sinistralité. Et enfin, il arrive que certains sinistres aient une une charge plus volatile, plus extrême que ce à quoi on peut s'attendre. Ces méthodes, quel que soit leur niveau de raffinement, ne sont pas capables de tenir compte de telles irrégularités.

C'est pourquoi nous décidons de nous tourner vers d'autres types de modèles, afin de faire une modélisation au niveau individuel, capable de quantifier la marge d'erreur existante pour chaque estimation faite, mais aussi capable de prendre en compte une variabilité temporelle ainsi que la réalisation et la modélisation d'événement rares ou extrêmes.

1.2.3 Les modèles de Micro Level reserving

Les développements récents de la littérature actuarielle dans le cadre du provisionnement individuel (c'est-à-dire en considérant dossier par dossier) sont réalisés dans un cadre stochastique.

Arjas [10] et Norberg [152][153] ont posé le cadre théorique en utilisant notamment des idées issues de la théorie des martingales qui reposent sur un cadre probabiliste plutôt que statistique.

Prédiction des engagements en cours

Norberg [152] [153] propose l'utilisation des processus de Poisson en temps continu. Le nombre de sinistres et la gravité de ces sinistres sont modélisés de façon séparée. À la survenance d'un sinistre, on associe à celui-ci un vecteur de variables aléatoires modélisant les différents éléments du cycle de vie du sinistre.

En 1999 [153], il corrigea le problème de la prévision des engagements en cours déclarées, mais non réglées en modélisant les paiements partiels par un processus Dirichlet.

Haastrup [91] reprendra cette idée en se basant sur une approche bayésienne, tandis que [124] décidera lui d'incorporer des informations sur le règlement qui peuvent gérer les effets saisonniers, les changements dans la composition de l'entreprise et les types de réclamations ainsi que les changements dans la composition de la taille des réclamations. Antonio & Plat [9] décideront d'adapter le modèle en modélisant le développement en temps continu des sinistres individuels.

Crevecoeur et Antonio [53] proposent d'utiliser un GLM pour modéliser les provisions en temps discret. Étant donné que les GLM sont appliqués dans la tarification des contrats d'assurance, l'idée est de combler l'écart entre la méthodologie de tarification et de provisionnement.

Les méthodes utilisant le Machine et le Deep Learning

L'émergence des méthodes de types arbres fin des années 1990 et l'apparition de technologies plus puissantes (telles que les cartes graphiques) a permis l'application de nouvelles méthodes statistiques au provisionnement individuel. Le rapport de Jamal [103], fait l'état de l'art des méthodes de Machine Learning est de leur utilisation en assurance.

Wüthrich (2018) [216] propose l'utilisation d'arbre de régression afin de modéliser le provisionnement individuel. Il propose aussi une adaptation du modèle de Mack en ajoutant de l'information individuelle et en utilisant des réseaux de neurones [217].

Les différentes variantes des méthodes d'arbres ont permis plusieurs versions de provisionnement reposant sur ces algorithmes. On peut penser à l'application du Delta Boosting de Lee[132] qui est une variante du GB. Ou à l'Extra-Tree [17] qui est une variante du CART.

Lopez et al. (2016) [138] proposent une méthode d'adaptation de l'algorithme CART aux données censurées en assurances. On y trouve une stratégie d'élagage des arbres de régressions intégrant la pénalisation via une pondération liée à l'estimateur de Kaplan-Meier. D'un point de vue mise en œuvre pratique, les principales étapes de calculs sont les suivantes :

1. calcul des poids de Kaplan-Meier.
2. construction de l'arbre.
3. élagage de l'arbre afin de l'adapter aux données censurées.

Carreau et Bengio (2009)[36]propose une estimation de la charge en utilisant un Mixture Density Network. Le principe du MDN (Bishop 1994 [25]) est de minimiser la loss de notre loi de mélange. Le réseau de neurones trouve donc les paramètres de notre loi de mélanges. Bien que principalement utilisé de base pour des mélanges gaussiens, ils proposent ici de l'adapter en appliquant un mélange Gamma-Pareto. Le problème des MDN est leur difficulté à converger et l'obligation d'utiliser des "astuces" pour obtenir la convergence du modèle [30].

1.3 La théorie des valeurs extrêmes (TVE)

La plupart des études empiriques et des modèles ne se concentrent que sur les propriétés moyennes de la distribution. La Théorie des valeurs extrêmes (TVE) a été développée pour l'estimation de la probabilité d'occurrence d'évènements extrêmes, elle permet l'extrapolation du comportement de la queue de distribution des données à partir des plus grandes observations. C'est dans cette optique qu'on cherchera à modéliser la loi du maximum.

En assurance non-vie, la Théorie des Valeurs Extrêmes (TVE) joue un rôle important, notamment pour couvrir le risque du portefeuille. La TVE vise à couvrir les évènements avec une occurrence rare (crack boursier, épidémie, attentat) mais aussi des évènements ayant une valeur importante.

On pense par exemple aux évènements climatiques tels que les ouragans, les tempêtes ou les tremblements de terre pouvant entraîner des pertes matérielles importantes, comme des locaux détruits ou brûlés (incendie), qui elles-mêmes peuvent conduire à une impossibilité pour les salariés de travailler durant une période plus ou moins longue, et c'est donc à l'assureur de prendre en charge toutes ces pertes.

Les applications sont très répandues et diverses [76] que ce soit en finance et assurance [77] [35] [82] [144], en ingénierie [37], en hydrologie[62] et également en médecine [202]. C'est pourquoi, nous nous tournons vers la TVE afin de modéliser le risque et l'aléa de ce type de sinistre.

Pour fournir des exemples pratiques plus proches de notre domaine d'application, suite à un accident de voiture, une victime pourra perdre son emploi, car elle sera dans l'incapacité physique de travailler. L'assureur devra donc prendre en charge ces pertes de gains professionnels futurs (PGPF) mais aussi les frais annexes.

Un sinistre extrême ne touche pas uniquement l'assureur au sens propre, mais aussi son secteur de réassurance qui va devoir le prendre en charge. Un seuil permettant de définir si un sinistre est grave ou non est nécessaire afin d'optimiser au mieux la gestion possible de ce sinistre. Celui-ci permettra également de déterminer quand le sinistre sera cédé à son secteur de réassurance, mais aussi de calculer au mieux la prime nette du contrat de réassurance.

Il existe deux principales approches en Théorie des Valeurs Extrêmes :

- La première repose sur les k plus grandes observations d'un échantillon et détermine les trois lois possibles des extremas (synthétisées sous le nom de loi GEV pour Generalized Extreme Value distribution que l'on présentera plus tard).
- La deuxième utilise les observations au-delà d'un seuil déterministe, c'est la méthode Peaks Over Thresholds (POT).

Dans cette partie, nous donnons un aperçu de la théorie probabiliste des valeurs extrêmes. Nous commençons par rappeler quelques résultats exacts concernant leur distribution. Puis, nous présentons le théorème des valeurs extrêmes qui est un théorème asymptotique. Nous précisons que nous

n'énoncerons que les principaux résultats de la Théorie des Valeurs Extrêmes sans démonstration. Nous vous invitons à vous référer aux deux ouvrages de Embrechts [66](1997) et Beirlant (2006) [18] pour plus de détails. Quant à son application au domaine de l'assurance automobile, Benlagha et al [22] l'utilisent afin de construire des classes homogènes en termes de sinistralité.

1.3.1 Notations

Soit n observations $(x_1, ..., x_n)$ qui sont des réalisations d'un n-échantillon $(X_1, ..., X_n)$ de variables aléatoires (v.a.) indépendantes et identiquement distribuées (i.i.d.), de fonction de répartition (f.d.r.) F continue ($F(x) = \mathbb{P}(X \leqslant x)$). On note Q la fonction quantile associée définie par :

$$Q(p) = \inf\{x : F(x) \geqslant p\}$$

Dans nos applications, X représente la charge d'un sinistre. Le support de la densité de probabilité est noté $[l, u]$.

La fonction de répartition est estimée par la fonction de répartition empirique :

Définition 10. *Fonction de répartition empirique :* $\widehat{F}_n(x) = \frac{i}{n}$ *si* $x \in [x_{i,n}; x_{i+1,n}]$*, tel que* $x_{i,n}$*, i-ème valeur de l'échantillon ordonné.*

De même, on peut définir l'estimateur de la fonction quantile :

Définition 11. *Fonction empirique de quantile :* $\widehat{Q}_n(p) = \inf\{x : \widehat{F}_n(x) \geqslant p\}$

On définit le point terminal de F noté x_F de la façon suivante : $x_F = \sup\{x : F(x) < 1\}$ et $F^{\leftarrow}(y) = \inf\{x \in R : F(x) \geqslant y\}$ où $F^{\leftarrow}$ désigne l'inverse généralisée de F.

On note également $M_n = \max(X_1, ..., X_n)$. On souhaite également introduire une propriété liée au maximum d'une distribution et sa fonction de répartition.

Propriété 1. *Si* $F(x) < 1$ *alors* $\mathbb{P}(M_n \leqslant x) \to 0$ *quand* $n \to \infty$.

Cette proposition implique que si la valeur de la f.d.r. en x est plus petite que 1, alors x n'est pas le maximum.

Le comportement moyen des variables est donné par le Théorème Central Limite (TCL).

Théorème 1. ***(Théorème Central Limite)*** *Soit* $X_1, X_2, ...$ *des v.a. i.i.d. telles que* $\mathbb{E}[X_1^2] < \infty$ *et* $\sigma^2 = Var(X_1) > 0$. *Alors,*

$$\sqrt{n}\left(\frac{\bar{X}_n - \mu}{\sigma}\right) \xrightarrow{\mathscr{L}} Z, \ quand\ n \to \infty$$

où $\mu = \mathbb{E}[X]$ *et* $Z \sim \mathscr{N}(0, 1)$.

Bien que Laplace [123] ait initié les travaux sur l'étude du cœur de la distribution, on note un développement ayant eu lieu en parallèle entre l'étude du comportement moyen (Smirnov 1935 [189],1952 [190], Cramér 1946 [52]), et celle du comportement extrême (Gumbel 1935[89],1955 [90], Fisher 1927 [69], Fréchet 1927[71]).

Nous avons donc d'un côté la TVE, qui s'intéresse au minimum ou au maximum des observations, et de l'autre le TCL, qui lui s'intéresse au comportement moyen des observations.

Dans notre cas, nous nous intéressons au comportement des valeurs extrêmes et à l'estimation des quantiles extrêmes.En effet, notre but est d'améliorer la vision que nous avons de la réserve, or des sinistres particulièrement coûteux par rapport aux valeurs "standard" peuvent peser lourd dans la constitution de certaines provisions. En particulier, dans le domaine de la responsabilité civile, les dommages engendrés peuvent être particulièrement conséquent, entraînant des écarts considérables entre les montants.

D'un point de vue descriptif, il existe plusieurs méthodes, tant algébriques que graphiques, pour détecter les valeurs extrêmes et étudier la fonction F.

1.3.2 Loi du Maximum

La théorie des valeurs extrêmes a pour but d'étudier la loi du maximum d'une suite des variables aléatoires réelles même si, et spécialement si, la loi du phénomène n'est pas connue.

À noter que les résultats énoncés ci-dessous sont aussi vrais pour le minimum. En effet, $\min(X_1, ..., X_n) = -\max(-X_1, ..., -X_n)$ ce qui implique donc que les résultats sont identiques, et que les formules et théorèmes sont valables tant pour le minimum que pour le maximum.

On démontre facilement que la fonction de répartition de la loi de M_n est la puissance n-ième F^n de F.

$$\mathbb{P}(M_n \leqslant x) = \mathbb{P}(X_1 \leqslant x, X_2 \leqslant x, ..., X_n \leqslant x)$$

Or comme les variables sont indépendantes,

$$\mathbb{P}(X_1 \leqslant x, X_2 \leqslant x, ..., X_n \leqslant x) = \mathbb{P}(X_1 \leqslant x) \times \mathbb{P}(X_2 \leqslant x) \times ... \times \mathbb{P}(X_n \leqslant x)$$

Et comme elles ont la même loi,

$$\mathbb{P}(X_1 \leqslant x) \times \mathbb{P}(X_2 \leqslant x) \times ... \times \mathbb{P}(X_n \leqslant x) = \mathbb{P}(X_1 \leqslant x)^n = F^n(x)$$

Les propriétés statistiques de M_n données par sa fonction de répartition, dépendent principalement des propriétés de $F(x)$ pour les grandes valeurs de x. Pour les autres valeurs de x, l'influence de $F(x)$ décroît rapidement avec n (cf. proposition 1). L'information la plus importante à propos des extrêmes est donc contenue dans les queues de la loi de X. D'après la formule, nous pouvons conclure quant à la forme de la loi limite de M_n en faisant tendre n vers l'infini. Nous trouvons que la fonction de répartition est nulle pour les valeurs de X inférieures à la borne supérieure u et égale à un pour les valeurs de X supérieures à u. La loi limite est donc dégénérée puisqu'elle se réduit à une masse de Dirac portée en u.

Les formules exactes de la loi du terme maximum et de la loi limite présentent toutefois un intérêt limité. En pratique, on ne connaît pas F ce qui implique qu'on ne connaît pas non plus la loi du maximum.

De plus, même si on faisait l'hypothèse de connaitre F, la loi de M_n ne serait pas calculable facilement. Dans le cas d'une loi Normale par exemple, la distribution ne possède pas d'expression analytique pour F. Sa puissance n-ème est donc difficile à calculer et conduit à de sérieux problèmes numériques pour de grandes valeurs de n et pour de grandes valeurs de x. Pour ces raisons, nous sommes conduits à étudier le comportement asymptotique du terme maximum M_n.

Le théorème de Gnedenko [84] (1943) fait appel aux notions de fonctions à variation régulière et normalisée, rappelons tout d'abord la définition de telles fonctions.

Définition 12. *Une fonction G est dite à variation régulière (à l'infinie) d'indice $\rho \in R$ si H est positive à l'infini (c'est-à-dire, s'il existe A tel que pour tout $x \geqslant A$, $G(x) > 0$ et si pour tout $t > 0$*

$$\lim_{x\to\infty} \frac{G(tx)}{G(x)} = t^{\rho}$$

Dans le cas particulier où $\rho = 0$, on dit que G est une fonction à variation lente.

En remarquant que si G est à variation régulière d'indice ρ alors $\frac{G(x)}{x^{\rho}}$ est à variation lente, il est facile de montrer qu'une fonction à variation régulière d'indice ρ peut toujours s'écrire sous la forme $x^{\rho}l(x)$, où l est à variation lente.

Théorème 2. *[172] l est une fonction à variation lente si et seulement si pour tout $x > 0$*

$$l(x) = c(x)\,\mathrm{e}^{\int_1^x t^{-1}\epsilon(t)}$$

où c et ϵ sont des fonctions positives telles que

$$\lim_{x\to\infty} c(x) = c \in]0, +\infty[\; et \; \lim_{t\to\infty} \epsilon(t) = 0$$

On peut noter deux choses :

- Si la fonction c est constante, on dit que l est normalisée.
- Soit G une fonction à variation régulière d'indice ρ. En utilisant le fait que $G(x) = x^{\rho}l(x)$, on en déduit que pour tout $x > 0$

$$G(x) = c(x)\,\mathrm{e}\left\{\int_1^x t^{-1}\rho(t)\right\}$$

où c et ρ sont des fonctions positives telles que

$$\lim_{x\to\infty} c(x) = c \in]0, +\infty[\; et \; \lim_{t\to\infty} \rho(t) = \rho$$

Pour trouver, une distribution limite non dégénérée, il nous est nécessaire de transformer M_n. L'étude du comportement moyen et du comportement étant liés, il est assez naturel de se tourner, comme pour le TCL, vers une normalisation de la variable aléatoire.

La variable M_n est ajustée à l'aide des suites a_n (supposé positif) et b_n. Nous supposons l'existence d'une telle séquence de coefficients (a_n, $n > 0$). Gnedenko (1943) fournit la version définitive du théorème des valeurs extrêmes qui spécifie la forme de la loi limite F_Y quand la longueur de la période sur laquelle on observe les extrêmes croissent indéfiniment. Ce théorème repose sur quatre hypothèses de base :

Théorème 3. ***(Fisher-Tippet-Gnedenko) [84]***

Soit X_1,...,X_n suite de v.a i.i.d, et $M_n = \max(X_1, ..., X_n)$.

On dit que $F \in D(G)$ s'il existe deux suites a_n (a_n>0) et b_n telles que la convergence suivante ait lieu : le théorème de convergence de la loi limite du maximum énonce que si

$$\mathbb{P}\left(\frac{M_n - b_n}{a_n} \leqslant x\right) = [F(a_n x + b_n]^n \to G(x)$$

où G est une distribution non dégénérée, alors G a une distribution des extrêmes généralisée.

$G_{\xi}(x)$ n'est autre que la Distribution des Valeurs Extrêmes Généralisée notée GEV (ou EVD pour Extreme Value Distribution) qui est définie comme suit :

- $\xi > 0$:

$$G_\xi(x) = \begin{cases} 0 \; si \; x \leqslant 0 \\ \mathrm{e}^{-x^{-\frac{1}{\xi}}} \; si \; x > 0 \end{cases}$$

- $\xi < 0$:

$$G_\xi(x) = \begin{cases} 1 \; si \; x \geqslant 0 \\ \mathrm{e}^{-(-x)^{-\frac{1}{\xi}}} \; si \; < 0 \end{cases}$$

- $\xi = 0$:

$$G_0(x) = \mathrm{e}^{\mathrm{e}^{-x}} \; \forall x \in R$$

ξ est l'indice des valeurs extrêmes, il est appelé indice de queue. Selon son signe, on distingue trois domaines d'attraction notés $DA(H)$ ou $DAM(H)$:

$\xi = 0$: Gumbel, pour les lois à queue à décroissance exponentielle (queue fine)

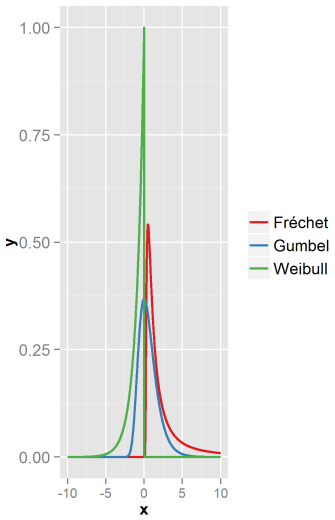
$\xi < 0$: Weibull, dans le cas ou X admet un point terminal fini (support borné)

$\xi > 0$: Fréchet, pour les distributions de type Pareto (queue lourde) ($1 - F(x) = x^{-\frac{1}{\xi}} l_{F(x)}$)

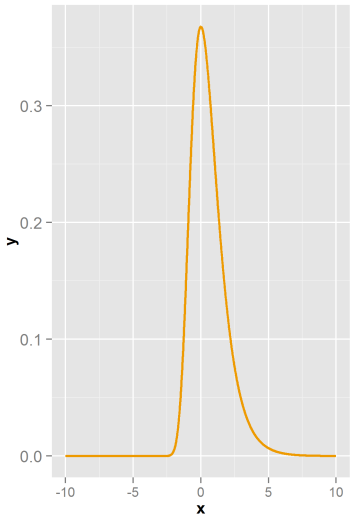
En d'autres termes, le maximum d'un échantillon de variables aléatoires i.i.d. après renormalisation ne peut converger en loi que vers ces trois lois s'il y a convergence. Ce théorème est très similaire au théorème central limite, à ceci près que le TCL s'applique sur la moyenne d'un échantillon de n'importe quelle loi ayant une variance finie, alors que théorème de Fisher-Tippet-Gnedenko énonce que si la loi d'un maximum normalisé converge, alors sa limite ne peut être qu'un certain type de loi. Il n'énonce pas que la loi d'un maximum normalisé converge. De plus, il donne des conditions nécessaires et suffisantes pour l'existence des constantes de normalisation. Quand la loi de la variable parente X est connue, ces conditions peuvent être utilisées pour déterminer le type de la loi limite.

Le théorème des valeurs extrêmes donne un résultat théorique concernant la loi limite des extrêmes. En pratique, il est difficile de trouver les suites a_n et b_n utilisaient pour la normalisation.

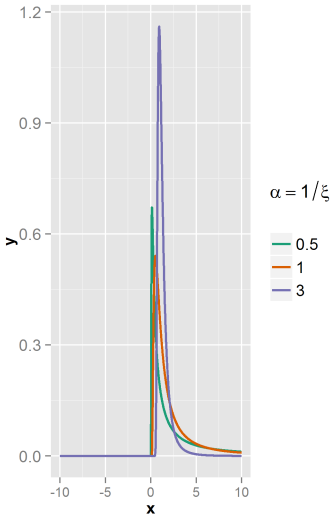
Notons également que l'on parle de domaine de Weibull, mais la loi associée est une reverse Weibull et non une Weibull.



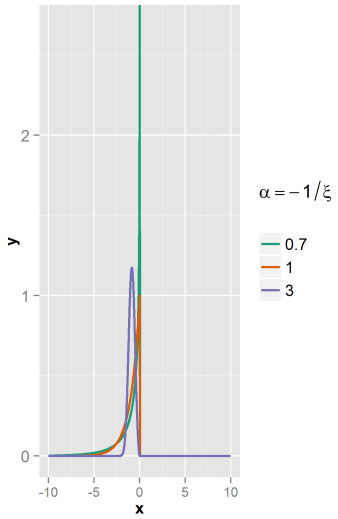
(a) Comparaison



(b) Gumbel



(c) Frechet



(d) Weibull

FIGURE 1.4 – Densités des lois de Gumbel et de Fréchet pour différentes valeurs du paramètre de queue et comparaison des densités des lois de reverse Weibull, Fréchet et Gumbel.

Pour en revenir au paramètre ξ, plus celui-ci est élevé, plus le poids des extrêmes dans la distribution initiale est important. On appelle ce type de distribution une distribution à « queue épaisse ». Plus généralement, on peut montrer que la loi de Gumbel est la loi des extrêmes pour les lois normales et log-normales, que la loi de Fréchet est associée aux distributions de Cauchy, Pareto et Student tandis qu'une loi uniforme a pour loi des extrêmes une loi de Weibull.

De plus, nous avons les deux propriétés suivantes sur le point extrémal de F.

Propriété 2. *Si x_F point extrémal de F, alors $M_n \to x_F$ en probabilité quand $n \to \infty$ (et même presque sûrement comme suite croissante.)*

La distribution asymptotique de M_n est donc dégénérée.

Propriété 3. - *$x_F < \infty \Leftrightarrow$ le support de F est borné à droite (loi uniforme, bêta, etc)*

- $x_F = \infty \Leftrightarrow$ *le support de* F *est non borné (loi normale, exponentielle, Gamma, Pareto etc)*

Cette proposition est intéressante, car elle nous permet, dans notre application, d'exclure directement une des trois lois vu précédemment. La loi de Weibull permet de modéliser les distributions dont le support de F est borné, s'il ne l'est pas, on utilisera la loi de Gumbel ou de Fréchet. Nos coûts étant considérés comme à support non borné, la charge n'admettant pas un point terminal fini, nous pouvons déjà exclure le domaine de Weibull.

De plus, [104] ont montré que dans le cas de données modélisées par un processus ARCH, la loi limite pour les extrêmes était la loi de Fréchet. Cette dernière loi apparaît donc comme étant a priori assez naturelle dans un contexte financier ou assurantiel.

Remarque : La représentation de Jenkinson-Von Mises [105] permet une écriture simplifiée de ces trois lois en une seule formule :

$$G_\xi(x) = \mathrm{e}^{-(1+\xi x)^{-\frac{1}{\xi}}}, \forall \xi \in R$$

définie $\forall x > -\frac{1}{\xi}$

C'est seulement par ces trois lois que nous pouvons exprimer la loi limite du maximum normalisé, d'où l'analogie avec le TCL.

1.3.2.1 Estimation de la loi d'extremum généralisée

Dans l'optique d'une prédiction de la charge à l'ultime, la modélisation de la GEV nous est plus que nécessaire. De plus, l'estimation du paramètre de queue nous sera quant à lui utile pour déterminer quels outils nous pourrons utiliser pour déterminer le seuil.

L'estimation de la GEV repose sur l'hypothèse que la loi limite est adaptée pour modéliser la loi à distance finie ; autrement dit que l'on a l'approximation suivante :

$$\mathbb{P}\left(\frac{M_n - b_n}{a_n} \geqslant x\right) \sim GEV(\mu, \sigma, \xi)$$

Cette hypothèse repose sur deux conditions :

- une valeur de n qui soit suffisamment élevée.
- le fait que l'inverse de la fonction de hasard $h(x) = \frac{1-F(x)}{f(x)}$ soit de dérivée h' quasi-constante, ce qui dépend de la distribution F.

Certes, les suites a_n et b_n dépendent de F, mais pour un n fixé, ce ne sont que de simples constantes permettant la normalisation de la suite $(X_1, ..., X_n)$. De plus, pour trouver les cinq paramètres de la formule du dessus, c'est-à-dire a_n, b_n, μ, σ et ξ, il nous suffit de trouver les paramètres de la GEV, μ, σ et ξ. Nous ne souhaitons pas aborder la partie mathématique, car ce n'est pas le but intrinsèque de cette thèse, c'est pourquoi nous invitons le lecteur à se référer au livre de [46] pour voir comment trouver les suites a_n et b_n.

Ainsi, pour modéliser la loi limite du maximum, il suffit de l'estimer directement par une loi $GEV(\mu, \sigma, \xi)$.

Pour estimer les paramètres de cette loi, on utilise la méthode des « Maxima par bloc ». Cette méthode consiste à subdiviser l'échantillon de n observations en K sous-échantillons de taille $\frac{n}{K}$, on dispose donc d'un échantillon de K maximas. Une fois les sous échantillons crée, on utilise l'EMV pour trouver les paramètres de la loi GEV [46].

Certes, il n'existe pas de solution analytique aux équations annulant les dérivants, mais on peut avoir recours à une solution numérique. Dans la plupart des cas, le calcul du maximum de vraisemblance ne pose pas de difficultés (Embrechts, Klûppelberg et Mikosch [66]) et l'estimateur conserve les propriétés classiques de l'Estimateur du Maximum de Vraisemblance (EMV) tant que $\xi > -\frac{1}{2}$ (Smith, [191]).

En revanche, Wong et Li [212] ont expliqué pourquoi il arrive parfois qu'en fonction de la taille et du nombre de blocs, l'EMV puisse ne pas converger. Cela est dû à la vitesse de convergence du modèle, donc la technique manque de robustesse.

La GEV étant une loi limite pour le maximum, il est donc nécessaire que les blocs soient suffisamment grands. Mais ceci nous pose un autre problème, car un bloc trop grand entraine une perte potentielle d'un maxima. Par exemple, si l'échantillon k contient trois valeurs supérieures au maxima du $k-1$, on a donc perdu deux maximas potentiels.

C'est pourquoi, pour compenser ce problème, on décide de modéliser les excès au-delà du seuil.

1.3.3 La loi des excès et la Loi de Pareto Généralisée

La GEV est réductrice, car l'utilisation d'un seul maxima implique une perte d'information continue dans les autres grandes valeurs de l'échantillon. La solution est donc de considérer plusieurs grandes valeurs au lieu de la plus grande.

La méthode *Peaks-Over-Threshold* (POT) utilise les observations qui dépassent un certain seuil, en particulier les différences entre ces observations et le seuil, appelées excès.

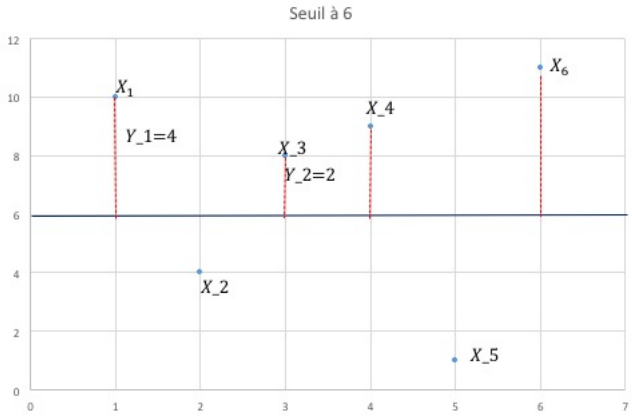


FIGURE 1.5 – La méthode P.O.T avec un seuil à 6

Il faut noter que le seuil déterminé ne doit être ni trop faible pour ne pas prendre en considération des valeurs non extrêmes, ni trop élevé pour avoir suffisamment d'observations.

Notons u ce seuil, il est défini de façon à ce que $u < z_F = \sup\{x : F(x) < 1\}$, z_F n'étant pas défini de façon aléatoire.

Définition 13. *Soit X une v.a. de f.d.r. F,*

$$\forall u < x_F \; F_u(x) = \mathbb{P}(X - u \leqslant x | X > u), x \geqslant 0$$

est appelée fonction de répartition des excès au-dessus du seuil u.

On remarque qu'on peut également noter F_u de la façon suivante :

$$F_u(x) = \frac{F(u+x) - F(u)}{1 - F(u)}$$

si $x \geqslant 0$ et 0 sinon.

De plus, on notera dans la suite, $Y = X - u$ pour $X > u$ les excès au-delà de u, et pour n v.a. observées $X_1, ..., X_n$, $Y_j = X_i - u$ tel que i est l'indice du j-ème excès et $j = 1, ..., N_u$.

L'une des dernières lois extrêmes que nous souhaitons introduire est la Loi de Pareto Généralisée (GPD). Nous l'utilisons afin de modéliser les excès au-delà du seuil de nos sinistres extrêmes. Pour cela, le théorème de Pickands [163](1975) est très utile lorsqu'on travaille avec des observations dépassant un seuil fixé puisqu'il assure que la loi des excès puisse être approchée par une GPD.

Théorème 4. *Théorème de Pickands [163] Si F appartient à l'un des trois domaines d'attraction de la loi des valeurs extrêmes, alors il existe $\sigma(u) > 0$ et $\xi \in R$ tels que*

$$\lim_{u \to x_F} \sup_{0 \geqslant y \geqslant x_F - u} |F_u(y) - G_{\xi,\sigma(u)}(y)| = 0$$

où $G_{\xi,\sigma(u)}(y)$) est la fonction de répartition de la loi de Pareto Généralisée et F_u f.d.r. des excès au-delà du seuil u.

De plus, quand u est grand, $F_u \approx G_{\xi,\sigma(u)}$.

La loi des excès ainsi que la GPD reposent sur l'utilisation d'un seuil noté u. Nous allons introduire dans les parties qui suivent des outils permettant de trouver la valeur de u.

1.3.4 Estimation du seuil

La théorie des valeurs extrêmes nous fournit des outils pour estimer la loi des observations au-dessus d'un seuil bien choisi et calculer les quantiles extrêmes à l'aide de cette loi estimée. Dans la suite, on s'intéresse au comportement (statistique) des observations au-delà d'un seuil u fixé (a priori u sera choisi « grand »).

Parmi les méthodes d'estimation de seuil disponibles, nous excluons volontairement la méthode "des valeurs record", qui est vraie asymptotiquement, mais qui donne des résultats aberrants dans notre cas (car nous avons un nombre fini d'observations).

Le Mean Excess Plot

L'estimation des paramètres de la distribution GPD pose le problème de la détermination du seuil u qui doit être suffisamment grand pour pouvoir appliquer les résultats de convergence en loi, mais pas trop grand afin d'avoir suffisamment de données pour obtenir des estimateurs de bonne qualité. Le Mean Excess Plot est le graphe qui a u associé $e(u)$ où $e(u)$ est la moyenne des excès au-delà du seuil u.

$$e(u) = E[Y | X > u]$$

Propriété 4. *Si* $(\gamma_1, ..., \gamma_n) \sim GPD_{\xi,\sigma}$, *alors* $\forall \xi < 1,\ E[\gamma | X > u] = \frac{\xi}{1-\xi}u + \frac{\sigma}{1-\xi}$.

Cette proposition nous permet de voir que dans le cas ou les variables suivent une loi GPD alors la MEF est une fonction affine en u.

On peut déterminer u en exploitant le résultat de la deuxième remarque. En effet, on sait que si l'approximation GPD est valide pour un seuil u_0 alors elle est valide pour tout $u > u_0$ (par la propriété de stabilité de la GPD). Donc pour $u > u_0$, la MEF est donc linéaire en u (voir proposition précédente). Pour déterminer u_0, on utilise donc le graphique de $e(u)$. Plus précisément, on trace $\widehat{e}(u)$ en fonction de u, avec

$$\widehat{e}(u) = \frac{\sum_{i=1}^{n}(X_i - u)+}{\sum_{i=1}^{n} \mathbb{1}_{X_i > u}}$$

Cela signifie que $e(u)$ est estimé par la somme des excès au-delà du seuil u, divisé par le nombre de points excédant le seuil u. De manière plus générale, le graphe de la Mean Excess Function permet d'avoir une bonne idée du comportement de la queue de distribution de X : lorsque le graphe de $e(u)$ est une constante, on a une distribution de type exponentielle, lorsque le graphe de $e(u)$ est une droite croissante, on a une distribution de type Pareto et lorsque le graphe de $e(u)$ est entre une constante et une droite croissante, on a une distribution à queue épaisse.

Avec la MEF, on voit que la loi exponentielle est un indicateur sur le domaine d'attraction de nos V.E. C'est pourquoi un graphique quantile-quantile indexé sur les quantiles de la loi exponentielle est utilisé pour déterminer graphiquement le domaine de nos V.E.

Le graphique quantile-quantile (QQ-plot)

La loi asymptotique du maximum normalisé de v.a. est de trois types possibles (Fréchet, Weibull et Gumbel), cette loi est indexée par un paramètre noté ξ. De même, la loi asymptotique des excès au-delà d'un seuil donné est une loi de Pareto généralisée indexée par le paramètre ξ et un deuxième paramètre σ. Ces paramètres, appelés respectivement indice de V.E. et paramètre d'échelle, apportent une information sur la forme de la queue de distribution de F.

On souhaite déterminer, pour un échantillon donné, la forme de la queue de distribution et avoir une idée du domaine d'attraction de la distribution. Le QQ-Plot est l'un des indicateurs graphiques (avec la Mean Excess Function qu'on verra plus tard) qui donnent une indication de l'appartenance probable à un des domaines d'attraction possibles.

Nous allons rappeler brièvement quelques informations sur la loi Exponentielle qui nous serons utiles pour la suite de la lecture :

Soit $X \sim \mathscr{E}(\lambda)$ alors la fonction de survie de X est $1 - F_\lambda(x) = \exp(-\lambda x), \forall x > 0$.

Propriété 5. *Soit* X *une variable appartenant au domaine d'attraction de la loi de Gumbel. Alors la loi des excès de* X *au-delà d'un seuil* u *converge vers une loi exponentielle lorsque* $u \to +\infty$

Le graphique quantile–quantile est une méthode nous permettant de mesurer graphiquement l'adéquation d'une variable observée à une loi théorique de fonction de répartition F continue. Lorsque X est à fonction de répartition F continue, $F(X) \sim U[0; 1]$. En notant $X_{(1)} \leqslant ... \leqslant X_{(n)}$ la statistique d'ordre associée à une variable, on a donc

$$(F(X_{(i)}))_{i=1,...,n} = (U_{(i)})_{i=1,...,n}$$

et donc en utilisant F^{-1}, on a l'équivalence suivante :

$$(X_{(i)})_{i=1,...,n} = (F^{-1}(U_{(i)}))_{i=1,...,n}$$

Le graphique quantile-quantile est donc donné par le couple

$$\{X_{(i)}, F^{-1}(1 - \frac{i}{n}), i = 1, ..., n\}$$

L'adéquation des observations à la loi de fonction de répartition F se traduit par la linéarité du nuage de points obtenu. C'est-à-dire que dans le cas de la loi exponentielle standard, notre cas :

$$\{X_{(i)}, -ln(\frac{i}{n}), i = 1, ..., n\}$$

Interprétation du QQ-plot En théorie, si les observations sont issues de la loi théorique, alors les points du graphique doivent se situer le long de la première bissectrice. En pratique, à l'exception de quelques points extrêmes, la plupart des autres points doivent se situer autour de cette droite.

Dans le cas de la TVE et de la loi exponentielle, on supposera u suffisamment grand ($u = u^*$ dans notre cas) et on distinguera trois cas en fonction de la loi dont sont issues les observations et du domaine d'attraction.

Gumbel : Les points du graphique sont approximativement alignés.

Fréchet : On observe une forme concave.

Weibull : On observe une forme convexe.

1.3.4.1 Les estimateurs non paramétriques de l'indice de queue

On présente ici trois estimateurs différents, tous basés sur les statistiques d'ordres obtenus à partir de la série initiale en considérant les k valeurs les plus grandes (ou les plus petites). k dépendra a priori de n, même si on ne le mentionnera pas dans la notation : l'idée est d'avoir $k \to \infty$ lorsque $n \to \infty$, mais sans prendre « trop »de valeurs de l'échantillon, ce qui conduit à imposer $\frac{k}{n} \to 0$.

Accessoirement, cela implique que se posera la question du choix optimal de k. En effet, il est indispensable de calculer ces estimateurs sur les queues de distribution. Choisir un k trop élevé engendre le risque de prendre en compte des valeurs non extrêmes, à l'inverse, un sous-échantillon trop petit ne permet pas aux estimateurs d'atteindre leur niveau de stabilité. On abordera ce point par la suite.

Enfin, on retiendra que l'approche non paramétrique n'est envisageable que si l'on dispose d'un nombre important d'observations : dans le cas où l'échantillon est de petite taille, il faut se tourner vers une approche non paramétrique que nous n'aborderons pas.

L'estimateur de Hill [98]

L'estimateur de Hill n'est utilisable que pour les distributions de Fréchet ($\xi > 0$) pour lesquelles il fournit un estimateur de l'indice de queue plus efficace que l'estimateur de Pickands. Il est défini par la statistique suivante :

$$\widehat{\xi}^H_{k,n} = \frac{1}{k-1} \sum_{j=1}^{k-1} \ln \left(\frac{X_{(j)}}{X_{(k)}} \right)$$

Si on choisit $k, n \to +\infty$ de sorte que $\frac{k}{n} \to 0$ alors on peut montrer que $lim_{k\to+\infty}\widehat{\xi}^{H}_{k,n} = \xi$ et l'estimateur de Hill est de plus asymptotiquement normal :

$$\sqrt{k}\frac{\widehat{\xi}^{H}_{k,n} - \xi}{\xi} \xrightarrow{\mathcal{L}} N(0,1)$$

Cet estimateur du maximum de vraisemblance est dans le cas particulier du modèle $S(x) = 1 - F(x) = Cx^{-\frac{1}{\xi}}$ une distribution de Pareto d'indice $\alpha = \frac{1}{\xi}$. Dans le cas général du domaine de Fréchet, la fonction de survie est de la forme $S(x) = 1 - F(x) = x^{-\frac{1}{\xi}}L(x)$ où L est une fonction à variation lente. Cela induit un biais important sur l'estimateur de Hill, qui est donc en pratique d'une utilisation délicate. Dans le cas général, la fonction L apparait comme un paramètre de nuisance de dimension infinie, ce qui complique l'estimation [24].

L'estimateur de Hill joue également un rôle dans la détermination du seuil u. Lorsque $\xi > 0$, l'estimateur de Hill est courant et assure un bon équilibre biais-variance selon Dress, de Haan et Resnick (1998) [170]. En notant $X_{(1)} \geq ... \geq X_{(n)}$ la statistique d'ordre, n le nombre d'observations, et k un entier inférieur ou égal à n, l'estimateur de Hill s'écrit :

$$\xi^{Hill}_{k,n} = \frac{1}{k}\sum_{i=1}^{k} ln\left(X_{(i)}\right) - ln\left(X_{(i)}\right)$$

Si $k \to \infty$ ($k/n \to 0$ as $n \to \infty$), l'estimateur est faiblement convergent. Sous des hypothèses supplémentaires sur k et sur la fonction de répartition, il est en outre asymptotiquement gaussien, ce qui conduit à tracer un intervalle de confiance sur le graphe de l'estimateur de Hill

$$IC_{95\%}(\xi) = \left[\widehat{\xi}^{Hill} - 1.96 \cdot \frac{\widehat{\xi}^{Hill}}{\sqrt{k}}\ ;\ \widehat{\xi}^{Hill} + 1.96 \cdot \frac{\widehat{\xi}^{Hill}}{\sqrt{k}}\right]$$

L'estimateur de Pickands [163]

Il est défini par la statistique :

$$\widehat{\xi}^{P}_{k,n} = \frac{1}{\ln(2)}\ln\left(\frac{X_{(k)} - X_{(2k)}}{X_{(2k)} - X_{(4k)}}\right)$$

Il présente l'intérêt d'être valable quelle que soit la distribution des extrêmes (Gumbel, Weibull ou Fréchet). La représentation graphique de cet estimateur en fonction du nombre k d'observations considérées montre un comportement en général très volatil au départ, ce qui nuit à la lisibilité du graphique. De plus, cet estimateur est très sensible à la taille de l'échantillon sélectionné, ce qui le rend peu robuste. Il est donc d'un maniement délicat. On peut noter qu'il est asymptotiquement normal, avec :

$$\sqrt{k}\frac{\widehat{\xi}^{P}_{k,n} - \xi}{\sigma(\xi)} \to N(0,1)$$

lorsque $k \to \infty$ la variance asymptotique étant donnée par :

$$\sigma(\xi) = \frac{\xi\sqrt{2^{2\xi+1}+1}}{2(2^{\xi}-1)ln(2)}$$

L'estimateur de Hill Alternatif

[171] Même si l'estimateur de Hill est bien moins volatile que celui de Pickands (quand on le représente en fonction du nombre d'observations considérées), il manque de précision et sa phase de stabilisation, qui devrait correspondre à l'indice de la valeur de queue, est parfois difficile à identifier. [171] ont mis au point une méthode très simple pour effectuer un « agrandissement » du démarrage du graphe de l'estimateur de Hill. L'idée est d'utiliser un changement d'échelle sur l'axe des abscisses pour passer en échelle logarithmique. Ainsi, la partie de courbe correspondant aux valeurs extrêmes est agrandie et plus précise. Au lieu de représenter graphiquement $\left\{k, \widehat{\xi}^{H}_{k,n}\right\}$, on trace $\left\{k, \widehat{\xi}^{H}_{n^{\theta},n}\right\}$ avec $0 \leqslant \theta \leqslant 1$. Le calcul du k optimal, qui détermine la séparation entre la queue de distribution et le reste des données, sera détaillé par la suite.

L'estimateur des moments ou de Dekkers-Einmahl et de Haan [55]

Cet estimateur est proposé dans Dekkers et al. Il est défini par la statistique :

$$\widehat{\xi}^{M}_{k,n} = 1 + \widehat{\xi}^{(1)}_{k,n} - \frac{1}{2}\left(1 - \frac{\left(\widehat{\xi}^{(1)}_{k,n}\right)^2}{\widehat{\xi}^{(1)}_{k,n}}\right)^{-1}$$

avec $\widehat{\xi}^{(i)}_{k,n} = \left(\frac{1}{k}\sum_{j=1}^{k} ln\left(\frac{X_{(j)}}{X_{(k-1)}}\right)\right)^{i}$. Cet estimateur est convergent et asymptotiquement gaussien :

$$\sqrt{k}\frac{\widehat{\xi}^{M}_{k,n} - \xi}{\sqrt{1+\xi^2}} \rightarrow N(0,1)$$

Comme pour l'estimateur de Hill, il n'est utilisable que pour le domaine d'attraction de Fréchet ($\xi > 0$).

Comparaison des différents estimateurs

Le comportement en termes de précision des différents estimateurs présentés a été illustré dans le cas d'une loi de Pareto par Planchet [165].

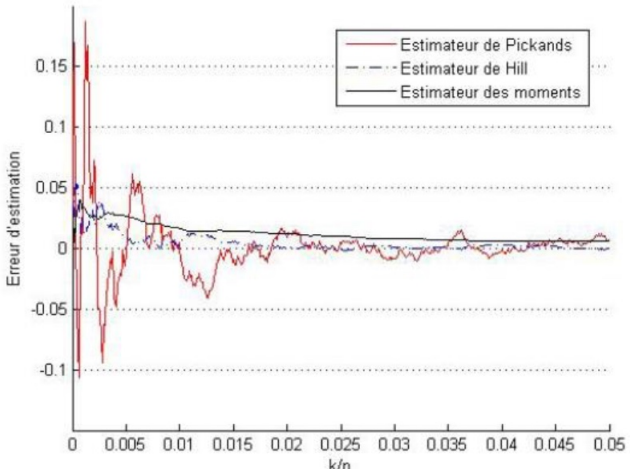


FIGURE 1.6 – Comparaison de l'erreur d'estimation en fonction de la méthode.

Il y démontre que l'estimateur de Pickands « paye » sa généralité par une moindre efficacité dans l'estimation de l'indice de queue. On y observe également la bonne performance de l'estimateur de Hill.

Une bonne modélisation des sinistres extrêmes assure à l'assureur un modèle de provisionnement. De plus, ces sinistres ont une durée de vie plus importante que la moyenne. Il y a donc une corrélation entre la durée de vie du sinistre et son montant, voir par exemple [136]. L'utilisation d'une telle information peut-être plus qu'utile pour l'assureur. De plus, lorsqu'on regarde notre base de données, le ratio de sinistre extrême clos/ ouvert est très important. Pour une victime grave dont le sinistre est clôturé, cinq sont ouvertes. Il est donc important de corriger ce phénomène de censure.

1.4 Les modèles de durée

Dans le cadre de cette thèse, nous avons eu affaire à des sinistres ayant une durée de vie longue : les sinistres responsabilité civile mettent en effet un temps considérable avant d'être stabilisés. Au cours de la vie du sinistre, beaucoup de choses peuvent évoluer comme l'avis de l'expert qui changera en fonction de l'état de santé de la victime. On peut également avoir des changements externes à la victime comme un changement de législation ou de gestion du sinistre. Il y a donc une volatilité liée à la durée du sinistre. L'étude de la variable de durée est donc une étape importante dans le cadre de la modélisation des sinistres graves [136] car ce sont souvent les sinistres ayant une durée de vie plus importante.

Pour pouvoir étudier ce type de sinistre, nous avons besoin d'avoir un historique qui soit suffisant. Plus la durée de vie du sinistre est importante, plus les risques de changement évoqués précédemment sont importants. C'est justement une des failles évoquée des méthodes Chain-Ladder.

L'assureur recevant des données de façon quasi continue, nous avons été obligés d'arrêter la collecte des observations à un instant t. En arrêtant la collecte de la donnée à une date fixe, nous avons deux possibilités pour nos observations. Soit nous possédons tout le déroulé de la vie du sinistre, soit l'information est incomplète. Dans ce cas-là, nous avons introduit un phénomène de censure.

L'utilisation d'un modèle de durée nous est utile pour modéliser à la fois la variable de durée mais aussi corrigé un biais de censure dans notre base en utilisant des méthodes de pondération.

Avant de présenter les modèles et les méthodes utilisées, nous souhaitons tout d'abord poser quelques notations qui nous seront utiles dans la suite de cette partie.

1.4.1 Quelques notions importantes

Posons tout d'abord le cadre dans lequel nous sommes. De par la construction de notre base, nous sommes en présence de censure à droite. On parle de censure à droite lorsque l'individu n'a pas subi l'événement au moment où il cesse d'être observé. En effet, les sinistres n'étant pas tous clos, nous n'avons pas pu observer toutes leurs durées de "vie". En revanche, on sait depuis combien de temps ils sont observés. C'est pourquoi pour la suite de cette partie, nous poserons, pour une observation i (correspondant à un sinistre) :

- son temps de censure C_i (temps qui s'est écoulé entre l'ouverture et la date où l'on cesse l'observation du sinistre sans qu'il soit clos)
- son temps de survie T_i (durée de vie du sinistre)
- la durée réellement observée $Y_i = min(T_i, C_i)$

. l'indicateur de censure $\delta_i = \mathbb{1}_{T_i \leq C_i}$

Avant de rentrer plus en détails sur l'utilisation des modèles de durée, il convient de définir les outils mathématiques utiles à l'étude des durées.

La fonction de survie joue un rôle important dans l'analyse de durée, c'est elle qui nous permet d'étudier la distribution de notre durée, mais qui est aussi au cœur de tous les modèles que nous évoquerons et que nous utilisons.

Soit U une v.a.r. positive continue représentant une durée, on définit $S_U(t)$ la fonction de survie de la façon suivante : $S_U(t) = 1 - F_U(t)$ où $F_U(t)$ est la fonction de répartition. Une autre notion importante est le taux de risque instantané de U au moment t, il représente, à une normalisation près, la probabilité que l'évènement U se réalise dans un intervalle de temps très petit après le temps t, sachant qu'il ne se soit pas produit avant t. Il est défini, dans le cas d'une variable absolument continue, par la formule

$$\lambda_u(t) = \lim_{h \to 0} \frac{\mathbb{P}(t \leq U \leq t + h | U \geq t)}{h} = \frac{S'_U(t)}{S_U(t)} = -ln(S_U(t))'$$

Du risque instantané, on peut définir le taux de risque cumulé, $\Lambda_U(t) = \int_0^t \lambda_u(s)ds$. En utilisant cette notation, on peut exprimer la fonction de survie de la façon suivante $S_U(t) = \mathrm{e}^{-\Lambda_U(t)}$. En assurance, étudier la fonction de survie, revient à étudier la proportion de sinistres qui seront ouverts pendant une durée supérieure à t. Intuitivement, utiliser un estimateur de la fonction de survie nous aiderait à corriger le biais introduit par la censure.

1.4.2 L'estimateur de Kaplan-Meier

L'estimateur de Kaplan-Meier [112] est un outil essentiel de l'analyse de durée. Il peut être vu comme l'extension de la fonction de répartition empirique au cas de la censure à droite. Notons que cet estimateur est très utilisé en assurance vie pour le calcul des tables de mortalité [166] [50] ainsi qu'en assurance non-vie pour la modélisation de coût [138].

La définition de l'estimateur de Kaplan-Meier de la fonction de survie S_T est

$$\widehat{S}_T(t) = \prod_{Y_i \leq t} \left(1 - \frac{\delta_i}{\sum_{j=1}^n \mathbb{1}_{Y_j \geq Y_i}}\right),$$

dans le cas où, pour simplifier, on a considéré qu'il n'y avait pas d'ex-aequos (c'est-à-dire que $Y_i \neq Y_j$ pour tout $i \neq j$) dans l'échantillon. Une formule plus générale peut être trouvée dans [112].

La consistance de l'estimateur de Kaplan-Meier, étudiée notamment par [81], [197] or [5], repose sur des hypothèses d'identifiabilité. On suppose que $\mathbb{P}(T = C) = 0$, et que T est C sont indépendants. La première hypothèse est impliquée par la seconde dès lors que T et C sont des variables continues. La justification de la nécessité de cette hypothèse pourra être perçue à travers la Proposition 6 ci-dessous.

À cette fonction de survie $\widehat{S}_T$ est associée une distribution, qui est l'analogue de la distribution empirique. Cette distribution est discrète, puisque l'on déduit de la définition de $\widehat{S}_T$ que cet estimateur est constant par morceaux. Les sauts se produisent uniquement aux observations non censurées. Contrairement à la distribution empirique dans le cas non censuré, cette masse dépend du rang de l'observation dans l'échantillon (pour la distribution empirique, la masse attribuée à chaque observation, c'est-à-dire l'amplitude du saut de la fonction de répartition empirique, est toujours de $1/n$).

En effet, si les observations censurées ne se voient pas attribuer de masse, la masse manquante est reportée sur les observations plus grandes (non censurées).

À noter que la distribution ainsi définie, si elle est toujours positive, n'est pas nécessairement une distribution de probabilité : si la plus grande observation est censurée, $\widehat{S}_T$ ne tend pas vers 0 quand t tend vers l'infini. Si T est presque sûrement finie, cette limite (donc cette "masse à l'infini") tend vers 0 avec n.

Il faut également noter les faiblesses bien connues de l'estimateur, notamment son instabilité à proximité de la queue de distribution. En effet, la convergence à la vitesse $n^{1/2}$ obtenue notamment par [5], repose sur l'hypothèse $\int \frac{dF_T(t)}{S_C(t)} < \infty$, qui est une hypothèse minimale pour que la variance asymptotique soit finie. Cette hypothèse traduit le fait que si la censure empêche d'observer suffisamment fréquemment la queue de distribution (i.e. S_C tend vers 0 très vite), il est difficile d'avoir suffisamment d'éléments pour estimer non paramétriquement la distribution dans sa partie droite.

Cette vision de Kaplan-Meier comme définissant une distribution est essentiel dans l'approche que nous développerons ultérieurement, car elle sera à la base d'une méthode de correction de biais causé par la censure. Cette méthode, appelée IPCW pour Inverse Probability of Censoring Weighting, est notamment exposée ci-dessous.

1.4.3 Les poids IPCW

Cette approche a été formalisée par [207], tout en ayant déjà été introduite par [219] et [199] auparavant. On peut également consulter [14] et [206]. On trouvera notamment des applications en reserving dans [138] ou [126].

À la base de cette approche, on peut remarquer (voir par exemple [185]) que l'estimateur de Kaplan-Meier peut s'exprimer

$$\widehat{S}_T(t) = \frac{1}{n}\sum_{i=1}^{n} \frac{\delta_i}{\widehat{S}_C(Y_i)} \mathbb{1}_{Y_i>t},$$

où $\widehat{S}_C$ désigne l'estimateur de Kaplan-Meier de C (i.e. on inverse les rôles de T et C dans la définition). À travers cette formule qui relie les sauts de l'estimateur Kaplan-Meier de T à l'estimateur de Kaplan-Meier de C, on peut interpréter $\widehat{S}_T$ comme une approximation d'un estimateur "idéal" (et inaccessible puisque dépendant de S_C), où le poids attribué à l'observation i serait $n^{-1}\delta_i S_C(Y_i)^{-1}$. La légitimité de cet estimateur idéal vient notamment de la proposition suivante, dont la preuve repose sur les hypothèses d'identifiabilité déjà citées.

Propriété 6. *Soit ϕ une fonction réelle de support inclus dans $[0, \tau[$, avec $\tau = inf\{t \geq 0 : P(C > t) = 0\}$ alors sous l'hypothèse*

$$H0 : P(C > T|T) = S_C(T)$$

, $\mathbb{E}\left[\frac{\delta\phi(Y)}{S_C(Y)}\right] = \mathbb{E}(\phi(T))$

De ce fait, on pourra estimer toute quantité du type $E[\phi(T)]$ par $n^{-1}\sum_{i=1}^{n}\delta_i\widehat{S}_C(Y_i)^{-1}\phi(Y_i) = -\int\phi(t)d\widehat{S}_T(t)$. Cette idée s'étend au contexte de la régression, où un vecteur aléatoire de caractéristiques X_i est disponible pour chaque observation. Néanmoins, l'utilisation du poids $n^{-1}\delta_i\widehat{S}_C(Y_i)^{-1}$ suppose alors deux hypothèses (voir par exemple [199]) :

- $\mathbb{P}(T \leq C|X, Y) = \mathbb{P}(T \leq C|Y)$,
- T indépendant de C.

Ces deux hypothèses sont simultanément satisfaites lorsque (T, X) indépendant de C quoique légèrement plus générales.

Ceci nous alerte sur un point de vigilance : dans notre application, lorsque nous utiliserons cette technique, cela suppose que la censure ne dépend pas des caractéristiques X. Typiquement, si C est la durée entre aujourd'hui (date de l'extraction) et la date d'ouverture du sinistre, X ne peut contenir de variables dépendant de la date d'ouverture. Voir notamment [136] pour plus de discussions sur le sujet. Des alternatives, plus complexes à mettre en œuvre, on été par exemple considérées par [126].

1.4.4 Le modèle de Cox

Les applications que nous considérons se situent dans le cadre de la régression (on cherche à expliquer ou prédire une variable à partir de caractéristiques X). Dans ce contexte, le modèle de Cox[50] est un modèle très populaire . C'est un modèle semi-paramétrique conjuguant flexibilité des hypothèses et simplicité d'interprétation. Il porte sur le taux de risque conditionnel, c'est-à-dire le taux de risque instantané de la distribution de $T|X$. L'objectif du modèle de Cox est donc d'évaluer l'effet de covariables sur la durée de vie d'un individu. Il appartient à la famille des modèles à hasard proportionnels.

Les modèles à hasard proportionnels sont définis par la relation suivante :

$$\forall t,\ \lambda(t|X) = \lambda_0(t)h(\beta, X)$$

où X est le vecteur des covariables, β le vecteur des coefficients et h une fonction positive. Les paramètres du modèle s'estiment par quasi-maximum de vraisemblance. On remarque que la fonction de hasard est le produit d'une fonction dépendant du temps $\lambda_0(t)$ et d'une fonction qui n'en est pas $h(\beta, X)$ On parle de modèle à risque proportionnel car,

$$\forall i, j\ \frac{\lambda(t|X_i)}{\lambda(t|X_j)} = \frac{h(\beta, X_i)}{h(\beta, X_j)}$$

Autrement dit, le rapport des fonctions de risque est indépendant du temps.

Dans le modèle de Cox, on suppose que $h(x)$ n'est autre que la fonction exponentielle.

$$\forall t,\ \lambda(t|X) = \lambda_0(t)\,\mathrm{e}^{(\beta' X)}\,.$$

À noter que ce modèle de régression permet de combiner plusieurs avantages. Tout d'abord, sa consistance est vérifiée même si la censure est autorisée à dépendre de X (car l'hypothèse d'identifiabilité nécessaire à sa convergence est T indépendant de C conditionnellement à X). Ensuite, son caractère semi-paramétrique permet capacité d'interprétation (l'effet des covariables se traduit dans le coefficient β), tout en ne nécessitant pas de faire l'hypothèse d'une famille de loi particulière.

1.5 Les Modèles Linéaires Généralisés

Le modèle linéaire généralisé (GLM) s'est imposé comme le modèle le plus utilisé dans les compagnies d'assurance pour la prédiction dossier-dossier, c'est-à-dire dans l'évaluation, sinistre par sinistre, du montant final de ceux-ci. Ces modèles sont également très utilisés pour déterminer la prime pure des assurances dommages de détail dont l'assurance auto représente la plus grande part.

Leur interprétabilité représente aussi un intérêt fondamental expliquant leur popularité. En effet, il est important pour l'assureur de justifier le choix d'un modèle interne auprès de l'ACPR, et un modèle simple comme le GLM facilite cette justification.

Bien entendu, cette simplicité empêche ces modèles de capturer des phénomènes complexes, contrairement aux modèles de machine learning considérés par la suite. Ils représentent néanmoins une sorte de référence auxquels les résultats doivent être comparés.

1.5.1 Hypothèses du modèle linéaire généralisé

Soit Y une variable à expliquer et X un vecteur aléatoire de variables explicatives, et $(Y_i, X_i)_{1\leq i\leq n}$ des copies indépendantes identiquement distribuées de ces variables.

Un modèle linéaire généralisé suppose deux hypothèses :

- La distribution de $Y|X$ appartient à une famille exponentielle donnée.
- De plus, $g(E[Y|X]) = \beta' X$, où g est dite fonction de lien, et β est un paramètre inconnu, g étant strictement monotone et connue.

Pour une mesure adaptée (par exemple mesure de comptage pour le cas discret ou Lebesgue pour le cas continu), la fonction de densité de la famille exponentielle considérée est de la forme

$$f(y_k, \theta) = \mathrm{e}^{\frac{y\theta - b(\theta)}{\phi} + c(y,\phi)},$$

où $\theta \in \mathbb{R}$ est le paramètre canonique de la famille exponentielle, et $\phi \in]0, \infty[$ est un paramètre de dispersion. Les fonctions b et c sont spécifiques de la distribution choisie. La fonction b sera toujours supposée deux fois dérivable de dérivée première inversible, dérivable et d'inverse dérivable.

Dans la famille des lois exponentielles, on retrouve Poisson, Bernoulli, Normale, Exponentielle et Gamma. Remarquons que nous considérons ici des paramètres θ réels et que pour les deux dernières lois citées qui admettent deux paramètres, c'est seulement l'un des paramètres ou une fonction des deux paramètres, qui correspond à notre θ, l'autre paramètre sera ici supposé connu et correspond à ϕ dans notre cas. À noter que si ϕ est inconnu, c'est ce qu'on appelle un paramètre de nuisance car il contient l'information sur l'amplitude du bruit et la variance.

En effet, une variable Y de densité $f(y, \theta)$ a pour espérance $E[Y] = b'(\theta)$, et pour variance $Var(Y) = \phi b''(\theta)$.

Loi	θ	$b(\theta)$	ϕ
$\mathscr{P}(\lambda)$	$log(\lambda)$	$\exp(\theta)$	1
$\mathscr{B}(1,p)$	$log(\frac{p}{1-p})$	$\log(1+\exp(\theta))$	1
$\mathscr{NB}(r,p)$	$log(p)$	$-r\log(1-\exp(\theta))$	1
$\mathscr{N}(\mu,\sigma^2)$	μ	$\frac{\theta^2}{2}$	σ^2
$\mathscr{E}(1,p)$	$-\lambda$	$-\log(\theta)$	1
$\gamma(a,b)$	$-\frac{b}{a}$	$-\log(\theta)$	$\frac{1}{a}$
$\mathscr{IG}(1,p)$[1]	$-\frac{1}{2\mu^2}$	$-\sqrt{-2\theta}$	$\frac{1}{\lambda}$

TABLE 1.2 – Quelques familles exponentielles

La fonction de lien canonique est telle que $g(E[Y|X]) = \theta(X)$, où l'on note $\theta(X)$ le paramètre correspondant à la loi de $Y|X$. Autrement dit, g est la bijection réciproque de b'. Cette fonction de lien canonique est à relier à des propriétés d'optimalité des estimateurs (efficacité).

1. Gauss Inverse

Loi	$\mathscr{P}(\lambda)$	$\mathscr{B}(1,p)$	$\mathscr{NB}(r,p)$	$\mathscr{N}(\mu,\sigma^2)$	$\gamma(a,b)$	$\mathscr{IG}(1,p)$
$g(x)$	$\log(x)$	$\log(\frac{x}{1-x})$	$\log(\frac{x}{x+r})$	x	$-\frac{1}{x}$	$-\frac{1}{2x^2}$

TABLE 1.3 – Fonctions de lien canonique.

Le modèle linéaire généralisé présente de nombreux avantages : bien que simple, il présente a priori deux éléments de flexibilité. D'abord, le choix de la famille exponentielle, qui permet de ne pas se reposer sur les moindres carrés pour estimer l'espérance conditionnelle. Ensuite, le choix de la fonction g adapté au problème. La réalité de sa pratique est souvent différente. Le choix de la famille exponentielle s'effectue dans une liste restreinte (nous citons les choix les plus classiques par la suite). Ensuite, une fois la famille choisie, la fonction de lien canonique (qui est parfois la seule directement implémentée dans les logiciels statistiques) est souvent effectué de façon automatique, sans vérifier que les données sont bien en adéquation avec celle-ci. Or, si les estimateurs du maximum de vraisemblance deviennent efficace dans ce cadre, ils deviennent asymptotiquement biaisés si le choix de la fonction g a été mal fait.

Par la suite, nous aurons à considérer essentiellement deux types de modèles linéaires généralisés : la régression logistique (pour prédire un label 1 ou 0, sinistre grave ou non grave), et la régression Gamma. Cette dernière vise à modéliser le montant d'un sinistre non extrême (le modèle est ensuite complété de l'analyse de la queue de distribution, via les approches de théorie des valeurs extrêmes).

1.5.2 Le modèle logistique

La régression logistique est largement utilisée en assurance pour les événements de type binaire. Par exemple, la modélisation d'un taux de transformation, d'un taux de résiliation, ou bien de la gravité ou non d'un sinistre. De plus, les trois cas précédemment cités sont au cœur même de l'activité de l'assureur, tant au niveau reserving qu'au niveau marketing.

Dans le cadre de la régression logistique binaire, la variable Y prend deux modalités possibles $0, 1$ (devis transformé ou non en souscription, grave/non grave pour un sinistre, mort/vivant pour une victime, résiliation ou non pour un contrat...)

Le modèle logistique repose en grande partie sur de la statistique bayésienne, et notamment sur la notion d'évidence définit par

$$E_v(p) = log\left(\frac{p}{1-p}\right)$$

Dans notre cas, p est la probabilité a posteriori d'un individu x_m d'être positif, $p = \mathbb{P}(Y = 1|X = x_m)$. Le LOGIT d'un individu x_m s'écrit

$$\begin{aligned} log\left(\frac{p}{1-p}\right) &= log\left(\frac{\mathbb{P}(Y=1|X=x)}{1-\mathbb{P}(Y=1|X=x_m)}\right) \\ &= log\left(\frac{\mathbb{P}(Y=1|X=x_m)}{\mathbb{P}(Y=0|X=x_m)}\right) \\ &= x_{m0}\beta_0 + x_{m1}\beta_1 + ... + x_{mn}\beta_n \\ &= x_m\beta' \end{aligned}$$

ou $x_m = (x_{m0}, x_{m1}, ..., x_{mn})$ et $\beta = (\beta_0, \beta_1, ..., \beta_n)$ le vecteur des paramètres.

Les estimateurs du vecteur des β_j qu'on note $\widehat{\beta_j}$ sont calculés en maximisant le log-vraisemblance du modèle. Pour en savoir plus, nous invitons le lecteur à voir la démonstration dans [85].

1.5.3 Loi Gamma

La loi Gamma est utilisée pour modéliser des distributions fortement asymétriques avec une décroissance rapide en queue de distribution. Elle est très utilisée pour la modélisation de la charge d'un sinistre.

Définition 14. *Nous appelons fonction Gamma la fonction définie par*

$$\Gamma(\alpha) = \int_0^{+\infty} t^{\alpha-1}\,\mathrm{e}^{-t}\,dt,\ \forall \alpha \in \mathbb{R}_+^*.$$

Définition 15. *Une variable aléatoire X suit une loi Gamma de paramètres α et β définie dans $\mathbb{R}_+^*$ si elle a pour densité de probabilité la fonction définie par*

$$f_X(t) = \begin{cases} \frac{\beta^\alpha}{\Gamma(\alpha)} t^{\alpha-1}\,\mathrm{e}^{-\beta t} & si\ 0 < t \\ 0 & si\ t \leq 0 \end{cases}$$

Si $\alpha = 1$, on retrouve une loi $\mathscr{E}(\beta)$.

Propriété 7. *Soit X une variable aléatoire suivant une loi Gamma de paramètres α et β définie dans $\mathbb{R}_+^*$ alors :*

-
$$\mathbb{E}(X) = \frac{\alpha}{\beta}\ \ et\ \ \ \mathbb{V}(X) = \frac{\alpha}{\beta^2}$$

- *Si $\alpha = \frac{n}{2}$ et $\beta = \frac{1}{2}$ alors $X \sim \chi^2_n$.*
- *Soit $X_i, i = 1, ..., n$ des variables aléatoires i.i.d. de loi $\mathscr{GA}(\alpha_i, \beta)$ alors $\forall\ c \in \mathbb{R}_+^*$ on a :*

$$\mathscr{GA}(c(X_1 + ... + X_n)) = \mathscr{GA}(\alpha_1 + ... + \alpha_n, \frac{\beta}{c})$$

De façon classique, les distributions Gamma sont utilisées pour modéliser des systèmes à coût "modéré". On séparera en général la base des sinistres en deux (en utilisant des techniques de choix de seuil comme en théorie des valeurs extrêmes), et la partie basse de la distribution sera analysée via des lois Gamma.

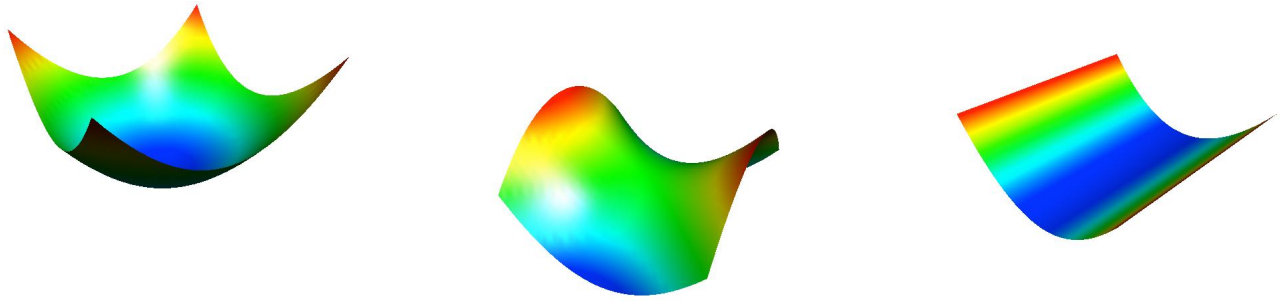
1.5.4 Régularisation

L'estimation du paramètre β dans les GLM s'effectue par maximum de vraisemblance. Lorsque la dimension de X devient importante (i.e. le nombre de covariables et/ou de leurs modalités est important), une réduction de dimension est nécessaire. Dans les cas que nous considérerons par la suite, cette dimension sera par moments de l'ordre voire supérieure au nombre d'observations.

Par ailleurs, lors de l'optimisation du critère que nous considérons, aucune contrainte n'étant appliquée au modèle, on peut obtenir des paramètres avec de grandes valeurs qui vont former un espace de solution très étendu. Ce type d'espace peut générer des vallées, ou ce qu'on appelle une selle de cheval (voir Figure 1.7). Le problème d'une telle typologie d'espace est que plusieurs vecteurs d'estimations peuvent aboutir aux mêmes solutions en termes de minimisation de l'erreur. La conséquence est qu'en cas de changement de nos données, on peut produire un modèle différent.

Les pénalisations que nous allons introduire permettent de palier ces différents problèmes afin de distordre notre espace. En anglais, on emploie le terme "shrinkage" car on rétrécit notre espace, on le ressert.

Nous allons présenter trois types de pénalisation, la pénalisation Ridge l_2 , Lasso l_1 et ElasticNet ($l_2 + l_1$), qui s'ajoute à la quantité à optimiser, c'est-à-dire la log-vraisemblance (que l'on cherche donc à maximiser).



(a) Un bol (b) Une selle de cheval (c) une gouttière

FIGURE 1.7 – L'espace des paramètres. Dans le cas ou notre espace est un bol, nous avons un minimum local qui est minimum global. Avec la selle de cheval, nous avons un minimum local qui n'est pas minimum global. Pour la gouttière, on a plusieurs solutions pour un même minimum.

1.5.4.1 La pénalisation Ridge

Cette pénalisation date des années 70 où [100] se base sur la norme l_2 du vecteur des paramètres, et est initialement proposé pour limiter l'instabilité liée à des variables trop corrélées entre elles. Cette pénalisation va donc diminuer la distance entre nos solutionx possible. Le terme de pénalisation est défini par :

$$-\lambda \sum_{i=1}^{n} \beta_i^2.$$

Dans ce premier cas, comme dans toutes les pénalisations que nous verrons par la suite, un hyper-paramètre λ doit être calibré afin de fixer la force de la pénalité. Cette détermination se fait, par exemple, par cross-validation.

1.5.4.2 La pénalisation Lasso

Lasso pour Least Absolut Shrinkage and Selection Operator [203], repose sur la norme l_1. La ou dans la Ridge on minimise la distance Euclidienne, avec la LASSO, on minimise la distance de Manhattan. La ou la Ridge va faire tendre les coefficienst vers 0, la Lasso va elle les fixer à 0. Elle élimine donc des variables (d'où le terme Selection Operator) Le terme de pénalisation est défini par :

$$-\lambda \sum_{i=1}^{n} |\beta_i|.$$

1.5.4.3 La pénalisation ElasticNet

L'idée de [221] repose sur l'intuition suivante : la pénalisation Ridge donne de meilleurs résultats, car elle détecte la colinéarité des variables, mais elle ne fait pas de sélection de variables. En revanche, si on combine les deux, on peut espérer cumuler les avantages. C'est ainsi que naît l'ElasticNet.

Le terme de pénalisation est défini par :

$$-\lambda \sum_{i=1}^{n} \left(\frac{1}{2}(1-\alpha)\beta_i^2 + \alpha|\beta_i| \right).$$

Le paramètre α est compris entre 0 et 1, et pour $\alpha = 1$, on retrouve la pénalisation LASSO.

Avant de clore cette section, nous aimerions faire une bref récapitulatif sur les différents avantages et inconvénients des méthodes que nous avons déjà citées.

Méthodes	Avantages	Inconvénients
Chain Ladder	Application simple Méthode de référence	Hypothèses fortes Sensibilités fortes des estimations des provisions
London-Chain	Application simple Moins contraint que Chain-Ladder	Sensibilités fortes des estimations des provisions
Mack	Même résultats que la méthode Chain Ladder Mesures de l'aléa associé	Hypothèses fortes Sensibilités fortes des estimations des provisions
GLM	Choix de la distribution des incréments Mesures de l'aléa associé	Hypothèses paramétriques Estimation des paramètres lourde

TABLE 1.4 – Tableau récapitulatif des avantages et inconvénients de chacune des méthodes

Le paysage de la modélisation actuarielle est en train de changer. Les assureurs délaissent les modèles statistiques économétriques classiques tels que les GLM afin de se tourner vers des modèles plus orientés Machine Learning. Cette révolution est en partie due à une diversification et une augmentation de la donnée. Les méthodes de Machine Learning ont l'avantage de se libérer de la restriction d'une distribution comme peut l'avoir le GLM. De plus elles sont aptes à capter des interactions entre les données et ainsi d'affiner l'appréhension du risque.

1.6 Les modèles de Machine Learning

Dans cette section, nous allons introduire les méthodes d'arbres. Nous commencerons par la base qui est l'algorithme de CART de Breiman (1984) [34] puis nous introduirons les Random Forests qui ont été crées par Breiman dans un but précis que nous expliquerons. Et enfin l'une des méthodes les plus utilisées sur Kaggle, l'algorithme du Gradient Boosting (Friedmann).

Ces méthodes ont l'avantage d'être utilisables tant pour des problèmes de régression, que pour des problèmes de classification, d'où le nom Classification And Regression Trees.

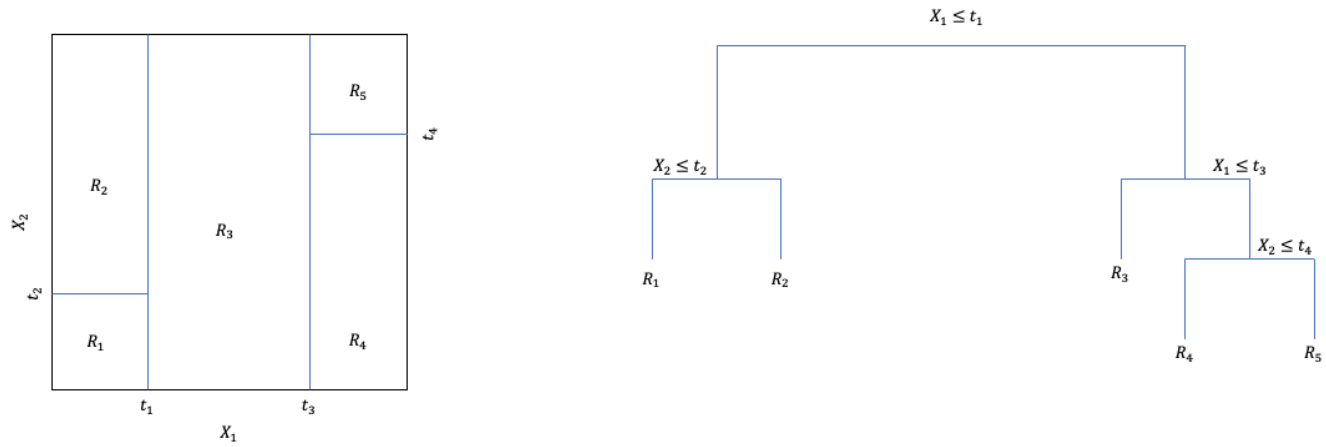
Dans le cas d'une classification, le résultat final est obtenu en faisant voter chaque arbre. Tandis que pour la régression, il est obtenu en moyennant le résultat des arbres.

Dans la suite de cette section, nous introduirons que très rapidement les algorithmes cités précédemment. Plus de détails peuvent être trouvés dans "The elements of statistical learning" de Friedman [74].

1.6.1 L'algorithme CART

Cet algorithme présente l'avantage de produire un modèle lisible et interprétable, tout en permettant une grande flexibilité. Il a notamment été utilisé en assurance pour l'élaboration d'un véhiculer ([179]), pour du provisionnement (voir section 1.2.3) ou pour du risque cyber [67].

Le but de cet algorithme est de partitionner l'espace de nos variables en des ensembles de rectangles (ou plutôt hyper-rectangles en grande dimension) afin de constituer des groupes d'individus les plus homogènes possibles du point de vue de la variable à prédire.



(a) La division de l'espace

(b) L'arbre de décision

FIGURE 1.8 – L'algorithme CART

De plus, le partitionnement se fait en prenant en compte une hiérarchie de la capacité prédictive des partitions intermédiaires. Cette hiérarchie permet de visualiser les résultats dans un arbre et de constituer des règles explicatives explicites orientées métiers.

Les arbres de classification et de régression présentent plusieurs avantages par rapport aux modèles linéaires :

- Ils autorisent les non-linéarités et les interactions.
- Ils sont susceptibles d'isoler des groupes de d'individus significatifs à partir d'un nombre restreint de règles de décision.

L'algorithme CART n'a pas que des avantages, il a un inconvénient bien connu, la faible stabilité du modèle. C'est notamment ce point qu'a voulu améliorer Breiman en créant les Random Forest.

1.6.2 Les méthodes ensemblistes

Les méthodes ensemblistes reposent sur la combinaison de plusieurs classifieurs faibles. Il est donc porteur d'un minimum d'information. Il existe plusieurs méthodes ensemblistes, les plus connues sont le bagging [31] et le boosting [60]. Et c'est sur elles que respectivement, les Random Forests, et le Gradient Boosting reposent.

1.6.2.1 Random Forest

Les arbres de décision sont très dépendants de l'échantillon d'origine : le simple ajout d'un individu peut fortement modifier le modèle ajusté.
Afin de contrer ce problème, [32] propose d'agréger plusieurs arbres de décision (constituant ainsi des forêts). Mais pour éviter d'avoir à chaque fois le même arbre si celui-ci était ajusté à partir du même échantillon, on introduit un tirage aléatoire des observations (Tree Bagging) et des variables (Feature Sampling) pour la construction de l'arbre.
On peut donc aisément dire que

$$Random\ Forest = Tree\ Bagging + Feature\ Sampling$$

1.6.2.1.1 Tree Bagging

Soit X une matrice d'individus avec n variables et m individus, et Y un vecteur de dimension m

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix} Y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{pmatrix}$$

La construction de B arbres de décision se fera comme suit :

- Tirer aléatoirement et avec remplacement B échantillons de (X, Y), notés (X_b, Y_b)
- Entraînement d'un arbre de décision sur le couple (X_b, Y_b) tiré précédemment.

Sur un jeu de test, on applique chacun des B arbres créés, puis il suffit de prendre la majorité parmi les B réponses.

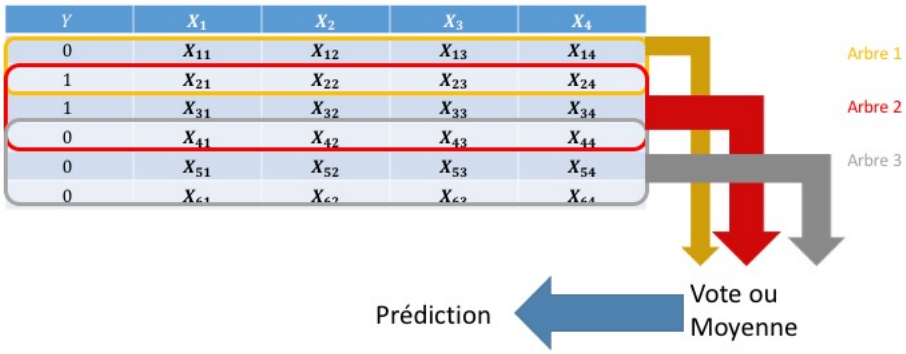


FIGURE 1.9 – Exemple d'un Tree Bagging avec trois arbres

1.6.2.1.2 Feature sampling

En plus du tirage aléatoire sur les lignes, le Random Forest, ainsi que d'autres méthodes de Machine Learning comme l'extra-tree ou le Gradient Boosting, qui sont aussi des méthodes ensemblistes s'appuyant sur les arbres de décisions, introduisent un tirage aléatoire sur les variables à utiliser.

Breiman suggère d'utiliser $\sqrt{n}$ pour un problème de classification et $\frac{n}{3}$ dans le cas d'une régression où n est le nombre de variables.[32]. Geurts [79] suggère d'utiliser toutes les variables dans le cas d'une régression. La valeur de ce paramètre dépendant avant tout du pouvoir prédictif des variables.

C'est le feature sampling qui est à l'origine de la diminution de la variance de l'ensemble créé.

Grâce à leur construction, les Random Forest présentent plusieurs avantages :

- Leurs constructions sont parallélisables.
- Sélection de variable via la méthode de permutation du feature sampling.

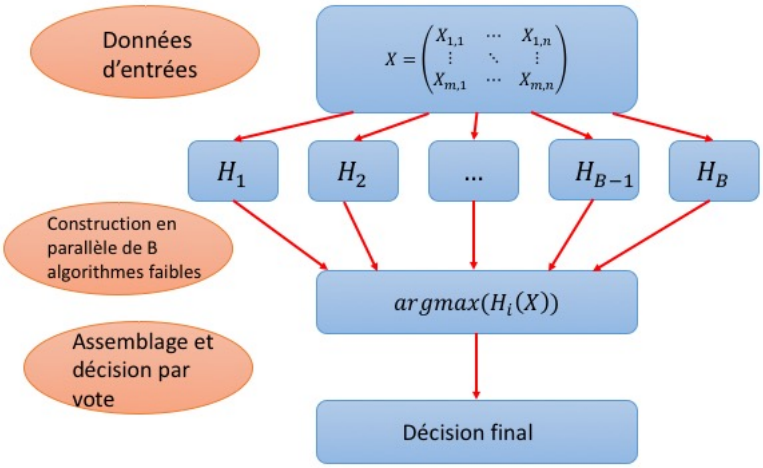


FIGURE 1.10 – Random Forest : processus de construction et d'assemblage

1.6.2.2 Gradient Boosting

Yoav Freund et Robert Schapire [73] introduisent Adaboost pour adaptive boosting qui n'est autre que l'ancêtre du Gradient Boosting [75]. Au début des années 1990, chacun de leur côté, Schapire puis Freund ont réalisé des travaux mettant en œuvre la notion d'algorithme faible (weak learning). Schapire [186] a démontré que n'importe quel algorithme faible pouvait être efficacement transformé (« boosté ») en un algorithme robuste. Un peu plus tard Freund [72] a présenté une nouvelle approche, boost-by-majority, qui améliore sensiblement l'approche initiale de Schapire. Mais l'idée commune des deux est qu'un méta-algorithme fait travailler successivement des algorithmes faibles chacun ayant accès à une distribution différente du problème se focalisant sur les observations difficiles à traiter et forçant ainsi son successeur à les traiter correctement. Le tout est assemblé dans un méta-algorithme. Pour faire simple, le terme boosting désigne, au sens large, les méthodes fonctionnant sur ce principe d'assemblage en série d'algorithmes faibles.

Là ou les Random Forest ont une construction du modèle en parallèle, le Gradient Boosting fonction lui en série. L'arbre K communique avec l'arbre $K-1$ et $K+1$. En effet, l'arbre $K-1$ communique

ses erreurs à l'arbre K qui va les corriger, mais il en commettra également. Et donc il communique les siennes à l'arbre $K+1$ et ce ainsi de suite.

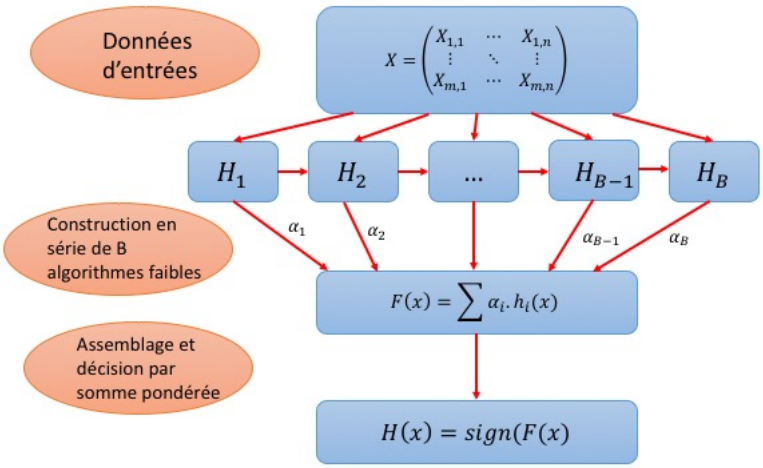


FIGURE 1.11 – Random Forest : processus de construction et d'assemblage

Les coefficients de pondération α_i dépendent uniquement des erreurs de chacun des h_i, notées ϵ_i :

$$\alpha_i = \frac{1}{2} \log \left(\frac{1 - \epsilon_i}{\epsilon_i} \right)$$

On remarque que plus l'erreur commise par un classifieur est faible, plus son coefficient sera important. Autrement dit :

$$\epsilon_i < \epsilon_j \Rightarrow \alpha_i > \alpha_j$$

Le Gradient Boosting n'est autre qu'un Adaboost généralisé à plusieurs fonctions de coût là où Adaboost n'en a qu'une seule. Cela est possible grâce à une descente de gradient [1] dans la construction itérative des algorithmes faibles.

Les modèles de Machine Learning sont de plus en plus mis en concurrence avec les modèles GLM pour de la provision dossier-dossier. Ils présentent l'avantage contrairement aux GLM d'être non paramétriques.

1.7 Introduction aux Réseaux de neurones

Les modèles linéaires généralisés ont été largement appliqués pour traiter des problématiques d'économétrie de l'assurance où le nombre de variables est relativement peu élevé, et leur structure est simple. Typiquement, en tarification, on dispose d'informations standards sur l'assuré, obtenues à la souscription, comme l'âge, le revenu, le lieu d'habitation etc.

Dans notre problème spécifique du provisionnement des sinistres graves, ce type d'information, s'il est utile, n'est pas suffisamment informatif pour parvenir à prédire l'issue d'un phénomène aussi

1. Voir Annexe

complexe. L'essentiel de l'information va se retrouver notamment dans des rapports d'expertise et/ou de gestionnaire de sinistre. Ces rapports sont des données textuelles, donc complexes à traiter (on parle de données non structurées). Afin de traiter ce type de données, les réseaux de neurones sont plus adaptés.

Bien que leur utilisation ait explosé que récemment, les réseaux de neurones sont antérieurs aux méthodes de type arbre. C'est en 1943 que McCulloch et Pitts ont proposé les premières notions de neurone formel [143]. Puis 15 ans plus tard, le premier modèle de réseaux de neurones est créé, c'est le perceptron [176] de Rosenblatt. Le problème de ces modèles, c'est qu'ils ont un coût computationnel important, et donc leur usage à l'époque était très limité. Avec l'émergence des technologies informatiques telles que les cartes graphiques ou les processeurs, leur utilisation est devenue possible et surtout généralisable.

Nous présenterons l'état de l'art des réseaux de neurones dans ce qui suit, mais nous vous invitons à vous rediriger vers [86].

1.7.1 Introduction au Deep Feedforward Networks

Les réseaux de neurones profonds (Deep Feedforward Networks en anglais) [101] sont une pièce maîtresse des modèles de réseaux de neurones. Ils ont pour but d'approximer au mieux une fonction f telle que $Y = m(\Theta, \mathbf{X})$ ou $\mathbf{X}$ est une matrice d'observation, Y la variable cible et Θ les paramètres du modèle permettant d'obtenir la meilleure fonction m.

Un neurone (voir Figure 4.1) est une fonction simple qui réalise une combinaison affine de ses variables d'entrée, puis applique au résultat une transformation non linéaire. Le neurone est donc caractérisé par des poids (w_i et b, b étant appelé terme de biais) et la fonction d'activation f, pour reprendre la notation de la Figure 4.1. Différentes fonctions d'activation classiques sont listées dans la table 1.5.

Au sein d'un réseau de neurones, on agrège et on compose des transformations des différentes variables d'entrée. La notion de "couche" permet d'établir une certaines hiérarchie entre les différents groupes de neurones (voir Figure 1.13). La première couche, dite "Input layer", correspond aux covariables. Celles-ci sont transmises directement aux neurones de la première couche. Les transformations effectuées par ces neurones sont transmis à de nouveaux neurones qui constituent la deuxième couche etc. Si l'on note Θ le vecteur contenant tous les paramètres des différents neurones du réseau, la dernière couche, dite "Output layer", contient $m(\Theta, X)$. Les couches intermédiaires sont appelées "Hidden Layers".

Pour un paramètre Θ fixé, le calcul de $m(\Theta, X)$ est appelé "Forward propagation", pour traduire le fait que l'information contenue dans les covariables traverse le réseau de la gauche vers la droite sans retour en arrière.

Du fait de la très grande dimension de Θ, une structure de réseau donnée peut produire une grande variété de fonctions. L'heuristique sous-jacente est que l'erreur d'approximation faite par ce modèle est faible, puisque l'ensemble de toutes les fonctions qui peuvent être produite par lui est très vaste. Toute la question est alors de trouver un Θ satisfaisant. Par satisfaisant, nous voulons dire que la fonction $m(\Theta, X)$ doit maximiser un critère donné (vraisemblance ou autre, en fonction du problème à résoudre) afin de déterminer sa qualité.

Cette optimisation peut se faire de façon efficace d'un point de vue algorithmique grâce à la structure des réseaux de neurones : la "backward propagation" (voir [181] [128]) permet le calcul rapide des dérivées partielles nécessaires à une descente de gradient.

La Descente de Gradient est un algorithme d'optimisation qui permet de trouver le minimum de n'importe quelle fonction convexe en convergeant progressivement vers celui-ci. L'objectif de la descente de gradient est de minimiser la fonction coût, qui justement est une fonction convexe. Nous détaillons plus en profondeur l'algorithme de Newton-Raphson en Annexe qui est un algorithme de descente de gradient.

Il existe trois façons différentes d'utiliser la descente de gradient [178] :

- Batch Gradient Descent [151]
- Stochastic Gradient Descent [29] [28] [129] [130]
- Mini Batch Gradient Descent [54] [48]

Dans le cas des réseaux de neurones, on utilise une descente de gradient appelée Mini Batch Gradient Descent : afin d'accélérer le calcul du gradient (par rapport à un calcul exact fait sur l'ensemble des données), on n'effectue ce calcul que sur des sous-ensembles de taille réduite (mini-batches). En moyenne, ces gradients calculés sur les mini-batches sont centrés (à une normalisation près) autour du gradient que l'on pourrait calculer sur l'ensemble des données (au prix d'un coût computationnel coûteux).

Voir le graphique 1.15 qui détaille les étapes d'optimisation des paramètres du modèle.

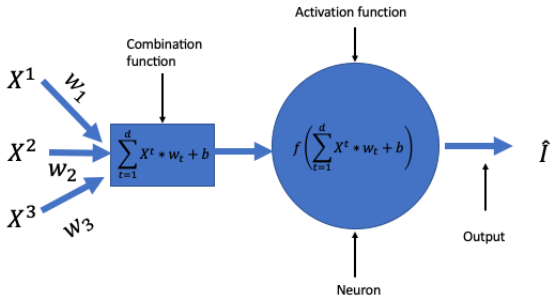


FIGURE 1.12 – Représentation d'un neurone simple.

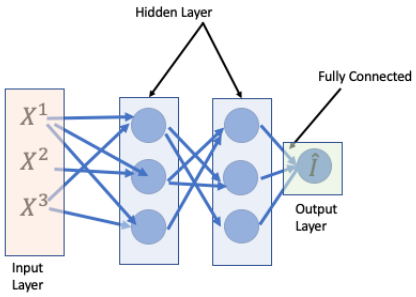


FIGURE 1.13 – Représentation d'un réseaux de neurones et de ses différentes couches. Chaque cercle est un neurone. Chaque rectangle est une couche composée d'un ou plusieurs neurones

Pour poursuivre avec les "conventions d'appellation", il existe deux métriques pour exprimer la taille / complexité d'un réseau de neurones. Soit on exprime le nombre de couches, soit son nombre de paramètres. Plus un modèle aura de paramètres plus on aura besoin de données et plus son apprentissage sera long.

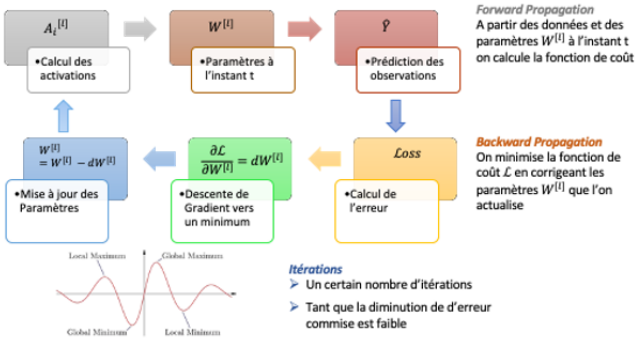


FIGURE 1.14 – Représentation des étapes d'un réseau de neurones. De l'étape forward à la backward.

Définition 16. *Une époque (epoch en anglais) est complète quand toutes les données ont fait une fois l'étape de forward et backward.*

La Mini Batch Gradient Descent est un compromis entre la descente de gradient stochastique [173] (ou on utilise une observation puis on met à jour le gradient) et la Batch Gradient Descent. Dans ce cas-là on utilise des minis batchs.

Définition 17. *Le Batch Size est le nombre total d'observation d'apprentissage présent dans un seul batch.*

Cette méthode de descente de gradient présente l'avantage d'avoir une mise à jour des paramètres de façon fréquente (comme le SGD) mais aussi d'avoir une convergence rapide (comme le Batch Gradient Descent). Un autre aspect plus "pratique" est qu'elle permet également de se contenter de charger partiellement les données en mémoire au fur et à mesure.

1.7.1.1 Les fonctions d'activation

Il existe plusieurs fonctions d'activation, mais les principales utilisées sont celles présentent dans le tableau 1.5.

Name	Function
Sigmoid function	$\sigma : \mathbb{R} \to]0,1]$, $x \mapsto \frac{1}{1+e^{-x}}$
Hyperbolic tangent	$tanh : \mathbb{R} \to [-1,1]$, $x \mapsto 2\sigma(2x)+1$
ReLU	$f : \mathbb{R} \to \mathbb{R}^+$, $x \mapsto max(0,x)$
Leaky ReLU	$f : \mathbb{R} \to \mathbb{R}^+$, $x \mapsto \alpha x \mathbb{1}_{x\leq 0} + x\mathbb{1}_{x>0}$ where α is a small constant
Swish function[169]	$\sigma : \mathbb{R} \to [0,1]$, $x \mapsto \frac{x}{1+e^{-x}}$

TABLE 1.5 – Les fonctions d'activation les plus utilisées

1.7.1.2 La pénalisation dans les réseaux de neurones

En plus des méthodes de pénalisation de la section 1.5.4 qui sont applicables tant à la fonction de coût qu'aux différents paramètres du modèle, il existe d'autres techniques de pénalisation telles que la Batch Normalization ou le Dropout Layer.

1.7.1.2.1 Batch Normalization

La formation des réseaux de neurones profonds est compliquée par le fait que la distribution des entrées de chaque couche change pendant l'entraînement, car les paramètres des couches précédentes changent. Cela ralentit la formation en exigeant des taux Learning Rate faible inférieurs et une initialisation soigneuse des paramètres. De plus la non-linéarité des fonctions d'activation peuvent causer la mort du gradient. Ce problème s'appelle le covariate shift. Pour le résoudre, Ioffe et Szegedy [102] proposent de normaliser les observations à l'entrée de chaque neurone.

Algorithm 1 Algorithme Batch Normalization

Require:

- Values of **x** over a mini-batch : $B = \{x_1 ... x_m\}$;
- Parameters to be learned : γ, β

$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$ // mini-batch mean
$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$ // mini-batch variance
$\widehat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$ // normalize
$y_i = \gamma\widehat{x_i} + \beta \equiv BN_{\gamma,\beta}(x_i)$ // scale and shift
return $\{y_i = BN_{\gamma,\beta}(x_i)\}$.

1.7.1.2.2 Dropout Layer

Srivastava et al[194] remarquent que bien que les réseaux de neurones soient très performants, le surapprentissage dans le cas de modèles très profonds est un vrai problème. De plus, de par le nombre de paramètres à estimer, un modèle très profond peut avoir un coût de computation très important.

C'est pourquoi ils proposent le Dropout Layer pour régler et le problème de sur-apprentissage et le problème de coût. L'idée clé est de supprimer au hasard des unités (ainsi que leurs connexions) du réseau neuronal pendant l'entraînement. Cela empêche les neurones de trop s'adapter. Au cours de l'entraînement, les observations parcourent un réseau "éclairci". Au moment du test, il est facile d'estimer l'effet de la moyenne des prévisions de tous ces réseaux éclaircis en utilisant simplement un seul réseau non éclairci qui a des poids plus petits. Cela réduit considérablement le sur-apprentissage et apporte un coût moins important de par le réseau "plus léger" lors de la forward propagation.

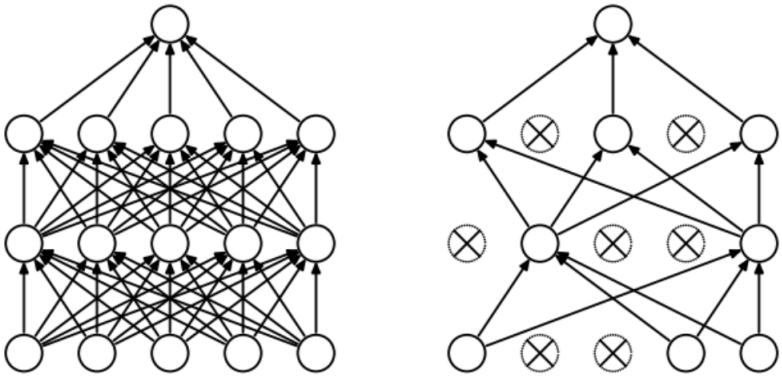


FIGURE 1.15 – Représentation du Dropout. Image de l'article "Dropout : a simple way to prevent neural networks from overfitting ."[194]

1.7.2 CNN : Convolutional Neural Networks

Introduits par Lecun au début des années 90 [127], les CNN sont principalement utilisés pour tout ce qui est traitement d'images (voir [120], [200], [83], [125]) Les CNN ressemblent au Deep Feedforward Networks, ils fonctionnent tous les deux en séries de processing et chaque processing est appelé un Layer.

Un CNN est composé de différents types de couches composant le réseau. On peut distinguer :

- Convolutional Layer : chaque neurone se concentre uniquement sur une partie localisée de l'entrée, voir ci-dessous ;
- Fully Connected Layer : chaque neurone de la couche est connecté à chaque neurone du suivant ;
- Normalization Layer : utilisées pour normaliser les entrées provenant de la couche précédente ;
- Padding et Pooling Layer : l'objectif est soit d'étendre, soit de réduire la dimension des entrées provenant de la couche précédente par des opérations simples, voir ci-dessous ;
- Output Layer : une prédiction de l'étiquette en combinant les résultats de la couche précédente.

De plus, là ou pour le Deep Feedforward Networks on aura un vecteur en input du modèle, pour le CNN on aura une matrice 2D ou 3D.

Bien qu'initialement destiné aux traitements de l'image, il est aussi depuis peu utilisé dans le cadre de la classification de texte ([113]) et c'est dans ce cas-là aussi que nous l'avons utilisé et comparé avec des méthodes plus classiques.

Convolutional Layer

La couche de convolution est la première étape du CNN. Elle extrait l'information de la donnée d'entrée tout en préservant l'information spatiale entre les coordonnées de celle-ci.

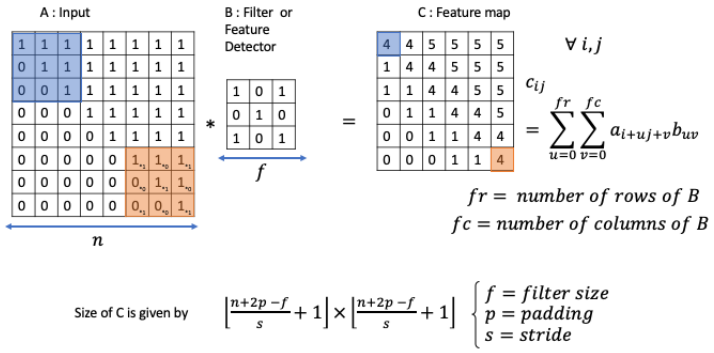


FIGURE 1.16 – Un exemple d'une couche de convolution. Chaque coefficient $c_{i,j}$ de la matrice de sortie est obtenue par combinaisons linéaires des coefficients $a_{i,j}$ dans la sous matrice de taille $f \times f$ de la matrice d'entrée. Le code couleur montre quel sous matrice est utilisée pour obtenir le coefficient correspond de la matrice de sortie C.

Dans la terminologie des CNN, la matrice B de la Figure 3.3 est appelée un filtre (ou kernel et parfois 'feature detector') et la matrice produite par la convolution entre la donnée d'entrée et le filtre est appelée 'Convolved Feature' ('Activation Map' ou 'Feature Map'). Le filtre et le coin en haut à gauche de la matrice d'entrée se superpose de façon à effectuer la première étape de la convolution. Puis, il se translate d'un pas s, appelé stride, de gauche à droite et une fois la ligne finie, il descend d'un pas s vers le bas. Chaque coefficient de C est une combinaison linéaire d'une sous matrice de A et du filtre f, c'est ce qu'on appelle l'opération de convolution. Chaque couche de convolution vise à capturer les dépendances locales dans l'entrée d'origine. De plus, des filtres différents font apparaître des caractéristiques différentes, et donc des structures différentes dans les données d'entrée. En pratique, un CNN apprend les valeurs de ces filtres par lui-même au cours du processus d'entraînement (bien que nous devions encore spécifier des paramètres tels que le nombre de filtres, la taille du filtre, l'architecture du réseau, etc.). Plus le nombre de filtres est élevé, plus on extrait de l'information, ce qui augmente la capacité du réseau à reconnaître des patterns cachés dans les données.

Pooling Layer

Les couches de convolution d'un CNN appliquent systématiquement des filtres appris aux données d'entrée afin de créer des feature map.

Elles s'avèrent très efficaces, et l'empilement de couches convolutionnelles dans des modèles profonds permet aux couches proches de l'entrée d'apprendre des caractéristiques de bas niveau et des couches plus profondes dans le modèle pour apprendre des caractéristiques d'ordre supérieur ou plus abstraites.

Le problème des filtres tient au fait qu'ils mémorisent l'emplacement de l'information dans la matrice d'entrée. Prenons l'exemple d'une photo, si l'objet que l'on souhaite identifier est en haut à gauche sur notre image d'entraînement, en le décalant à en haut à droite, on obtiendrait une feature map différente.

C'est pour palier ce problème qu'on utilise le Pooling Layer. Cette couche du modèle va créer une nouvelle matrice, plus petite, contenant que l'information importante pour le modèle. Il réduit donc la dimension de la matrice d'entrée, mais aussi le nombre de paramètres du modèle et donc sa complexité d'apprentissage.

En raison de la réduction agressive de la taille de la représentation, une possibilité est d'utiliser de petits filtres [88]. Bien que non obligatoire au fonctionnement du CNN [193] il est préférable de l'utiliser afin de réduire le risque de sur-apprentissage.

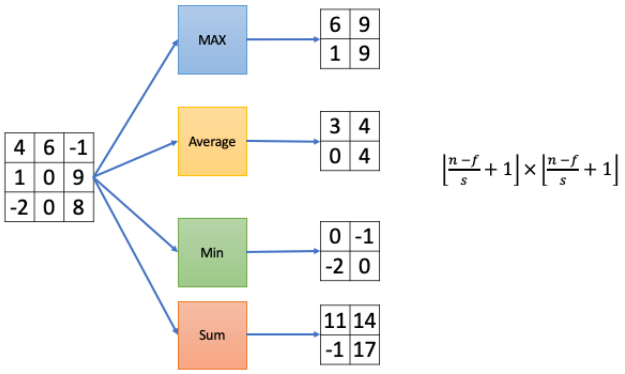


FIGURE 1.17 – Un exemple d'une couche de Pooling. Chaque fonction est appliquée à une sous matrice de taille 2x2

Padding Layer

La convolution pose plusieurs problèmes. Le premier, les éléments se trouvant au bord de la matrice sont convolés une seule fois contre les éléments du centre qui subissent plusieurs fois l'opération. On va donc tirer plus d'informations des éléments centraux que des éléments se trouvant au bord. Le deuxième, c'est que la convolution concentre l'information dans une dimension plus petite.

Pour éviter ces problèmes on utilise le Padding. Cela consiste à agrandir notre matrice en y rajoutant des zéros sur les bords.

On l'utilise aussi dans le cas où toutes les données n'ont pas la même dimension, on applique donc du zéro-padding afin de combler la différence entre la taille d'une observation et la taille maximum observée.

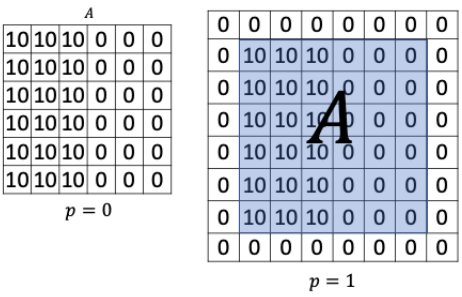


FIGURE 1.18 – Un exemple d'une couche de Padding.

1.7.3 RNN : Recurrent Neural Networks

Les réseaux de neurones récurrents (RNN) (voir [180], [159], [45], [107], [108]) répondent à un besoin spécifique : les données séquentielles. Dans le cas d'une phrase, l'ordre des mots est important, on ne peut pas échanger la phrase sans changer sa signification ou pire faire un non sens.

Là où les deux autres architectures traitent la donnée comme étant indépendante l'une de l'autre, le RNN va lui la traiter comme une séquence d'information.

Les RNN sont notamment utilisés pour tout ce qui est modélisation de texte (voir [146]) ou série temporelle (voir [47])

Dans le graphique 1.19, on parcourt donc successivement les entrées $\mathbf{X}^1$ à $\mathbf{X}^t$. À l'instant t, la $t-eme$ cellule combine l'entrée courante $\mathbf{X}^t$ avec la prédiction au pas précédent h_{t-1} pour calculer une sortie h_t. On obtient donc une relation de récurrence (d'où le terme Recurrent) $h_t = f(\mathbf{X}^t, h_{t-1})$

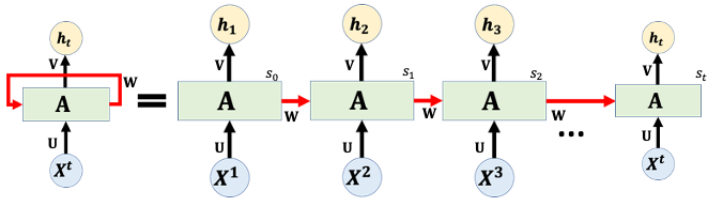


FIGURE 1.19 – Un RNN déroulé

T représente la taille de la séquence et M la taille de la représentation vectorielle de chaque élément de la séquence (M est égale soit à la taille du dictionnaire soit à la dimension de notre Embedding Matrix voir Chapitre 2). Une couche de type RNN prend en entrée une séquence de ce tableau (donc une matrice de taille (T, M)) et retourne en sortie une séquence de taille R qui est une représentation "compressée" résumant l'information reçue en entrée par le modèle. R est un hyperparamètre à choisir judicieusement, il représente le nombre de neurones par cellule de RNN.

Le graphique 1.20 montre le comportement de la $t - eme$ cellule L'entrée $\mathbf{X}^t$ est concaténée avec h_{t-1}. La fonction d'activation classique utilisée pour les cellules du RNN est la tangente hyperbolique $tanh$. On peut donc réécrire la formule de sortie :

$$h_t = tanh(\mathbf{W}.concat(\mathbf{X}^t, h_{t-1}) + \mathbf{b})$$

où $\mathbf{W}$ est une matrice de pondération de taille est une matrice de taille $(R + M, R)$ et $\mathbf{b}$ le vecteur de biais de taille R

La matrice $\mathbf{W}$ est partagée entre toutes les cellules d'une couche RNN. Autrement dit, c'est exactement la même fonction (avec les mêmes poids) qui est appliquée à chaque pas de temps t. On a donc une modélisation homogène de la séquence et on évite ainsi d'apprendre T matrices $\mathbf{W}$.

Bien que très performants sur les données séquentielles, les RNN présentent deux limites :

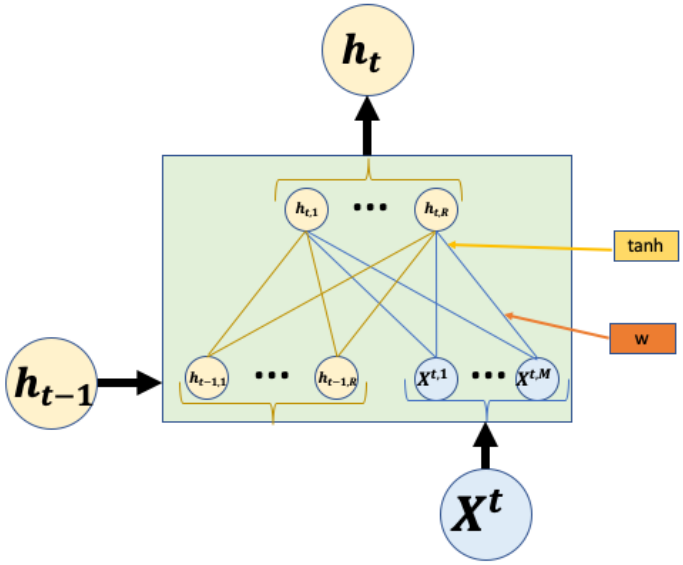


FIGURE 1.20 – Une cellule de RNN

Limite 1 : Un modèle à mémoire courte

Une couche RNN est une succession de cellules, chacune prenant en entrée la représentation de l'élément de la séquence ainsi que la sortie de la cellule précédente. Ces données d'entrée sont transformées, en passant notamment par une fonction tangente hyperbolique.

La tangente hyperbolique a tendance à écraser les valeurs qu'elle prend en entrée. En effet, elle est définie sur l'espace des réels, mais ses valeurs de sortie sont dans l'intervalle $]-1, 1[$. Ainsi, après un premier passage par la tangente hyperbolique, on obtient une valeur entre -1 et 1. Ensuite, comme $tanh(1) = 0.76$ et $tanh(-1) = -0.76$, un deuxième passage par la fonction $tanh$ résulte en un intervalle de valeurs encore plus réduit, et ainsi de suite.

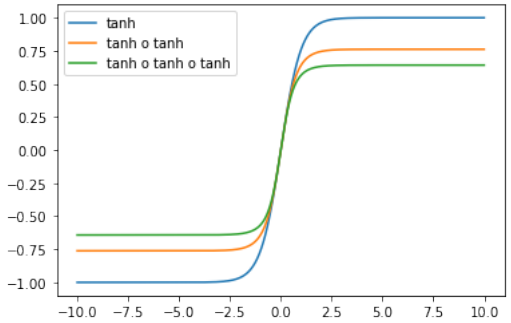
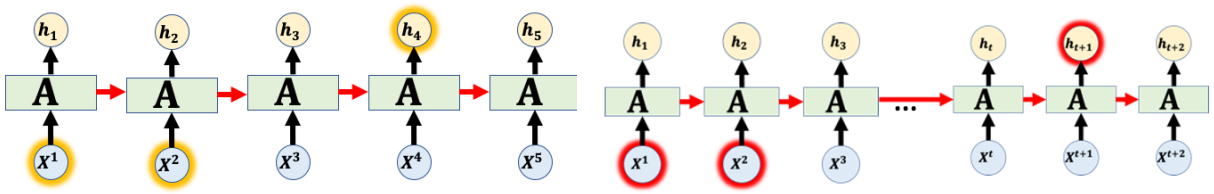


FIGURE 1.21 – La fonction $tanh$ et ses composées

Les informations provenant de $\mathbf{X}^1$ passent donc T fois à travers une tangente hyperbolique, là ou celles provenant de $\mathbf{X}^t$ n'y passent qu'une fois. L'écrasement de l'information est d'autant plus prononcé que la séquence est longue, autant sur une courte séquence les RNN arrivent à garder et transmettre l'information autant quand la séquence devient longue on perd toute information. L'information portée par $\mathbf{X}^1$ ne jouera pas dans la prédiction finale. Il est donc très compliqué pour un RNN basique de modéliser des séquences longues. Cet effet d'écrasement ne pourrait pas être

compensé par un poids plus fort associé à h_1, via la matrice de poids : comme la matrice est partagée par toutes les cellules de la couche, toutes les sorties intermédiaires sont pondérées par les mêmes poids.



(a) Séquence courte (b) Séquence longue

FIGURE 1.22 – Illustration de problème de mémoire du RNN

Limite 2 : Un problème de gradient

Un autre problème dû à la fonction $tanh$ et à ses composition est le problème de vanishing gradient (voir [20] [158]).

La fonction du réseau de neurones étant une composée, calculer son gradient revient donc à dériver des fonctions composées.

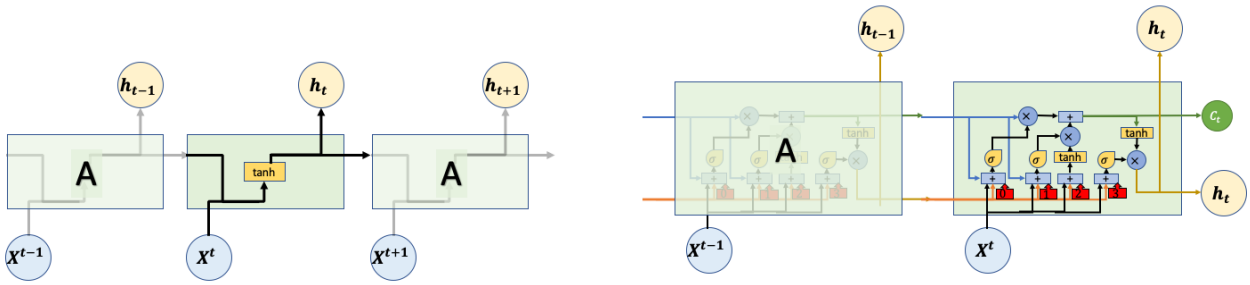
Ainsi, si notre modèle est composé de trois couches et est représenté par la fonction $f \circ g \circ h$, son gradient par rapport à W sera $\frac{\partial f}{\partial g}\frac{\partial g}{\partial h}\frac{\partial h}{\partial W}$.

Or la dérivée de la fonction $tanh$ est comprise entre 0 et 1, et plus on multiplie des valeurs dans cet intervalle entre elles, plus le résultat se rapproche de 0. Le gradient prend donc des valeurs très petites lorsque les séquences sont longues. ce qui implique que la mise à jour des paramètres devient donc très lente et l'entraînement du modèle est mis à mal.

LSTM : Long Short-Term Memory

Plusieurs variantes aux RNN standards ont vu le jour pour remédier aux problèmes évoqués précédemment. Les Gated Recurrent Unit [43] [44] et les Long Short-Term Memory [99] c'est cette dernière architecture que nous allons décrire.

L'idée des LSTM est de diviser le signal entre ce qui est important à court terme à travers le hidden state (analogue à la sortie d'une cellule de RNN simple), et ce qui l'est à long terme, à travers le cell state, qui sera explicité plus bas.



(a) Cellule RNN (b) Cellule LSTM

FIGURE 1.23 – Comparaison d'une cellule d'un simple RNN vs une cellule de LSTM.

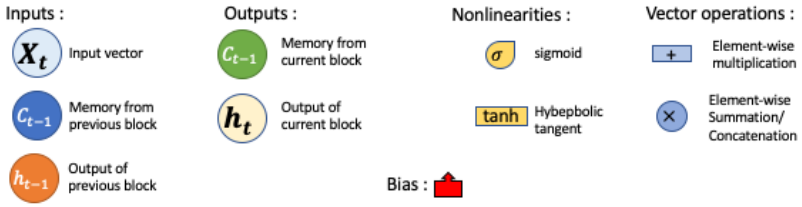


FIGURE 1.24 – Notation du LSTM

La force des LSTM réside dans la flèche de contexte illustré dans le graphique 1.25. C'est elle qui permet la circulation et la "protection" de l'information tout au long du traitement de la séquence.

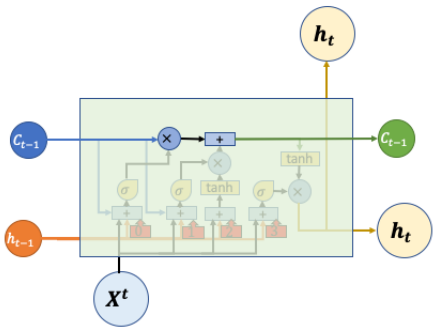


FIGURE 1.25 – Représentation de la flèche de contexte qui fait le lien entre l'information précédente, l'actuelle et la future.

De plus, l'utilisation de fonction sigmoïde dans son architecture permet aux LSTM de supprimer ou de mettre à jour l'information. On appelle souvent cela une porte (gate en anglais) et nous l'avons représenté au graphique 1.26. L'avantage de la sigmoïde est que ses valeurs sont comprises entre

0 et 1, il est donc facile de les interpréter comme un pourcentage d'ouverture de porte. Quand la sigmoïde tend vers 0, on ne met pas à jour l'information. A l'inverse, quand elle tend vers 1 on garde toute l'information apportée par un élément de la séquence.

Un LSTM possède trois de ces portes, pour protéger et contrôler l'information qui circule.

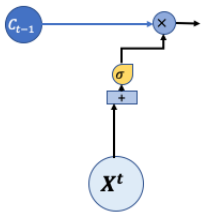


FIGURE 1.26 – Représentation d'une porte à l'aide d'une fonction sigmoïde

Comme le RNN, le LSTM définit donc une relation de récurrence, mais utilise une variable supplémentaire qui est le cell state noté C_t

$$h_t, C_t = f(\mathbf{X}^t, h_{t-1}, C_{t-1})$$

L'information passée est donc portée par deux vecteurs, celui du contexte C_{t-1} et comme pour le RNN, h_{t-1}. Ils rentrent ensuite en interaction avec l'élément courant $\mathbf{X}^t$.

Une cellule de LSTM prend en entrée 3 vecteurs :

- L'élément courant de la séquence $\mathbf{X}^t$ (vecteur de taille M)
- Le hidden state de la cellule précédente h_{t-1} (vecteur de taille R)
- Le cell state de la cellule précédente C_{t-1} (vecteur de taille R).

C'est le cell state qui s'occupe de préserver l'information. Pour éviter le problème du vanishing gradient, il est mis à jour de façon additive à chaque étape, sans passer par une activation.

La première étape du LSTM est la forget gate, c'est une couche sigmoïde dont la sortie est un vecteur de taille R ou ses valeurs sont comprises entre 0 et 1. À partir de h_{t-1} et $\mathbf{X}^t$ elle produit un vecteur f_t défini par la fonction

$$f_t = \sigma(W_f[h_{t-1}, \mathbf{X}^t] + b_f)$$

Cette porte agit comme un filtre pour « oublier » certaines informations du cell state. En effet, plus tard on mettra à jour la cell state, et durant la mise à jour, on multipliera terme à terme f_t et c_{t-1}, ce qui a tendance à annuler les composantes de c_{t-1}.

L'input gate est très similaire à la forget gate, elle a les mêmes entrées et produit, elle aussi, un filtre de taille R. En parallèle, un vecteur $\tilde{C}_t$ de taille R est créé par une couche $tanh$. C'est le vecteur candidat pour mettre à jour le cell state.

$$i_t = \sigma(W_i[h_{t-1}, \mathbf{X}^t] + b_i)$$
$$\tilde{C}_t = tanh(W_C[h_{t-1}, \mathbf{X}^t] + b_C)$$

La troisième étape consiste à mettre à jour la cell state. Pour cela, on utilise la forget gate pour oublier une partie de l'information passée et on y ajoute une partie de la nouvelle information.

$$C_t = f_t * C_{t-1} + i_t \tilde{C}_t$$

Enfin, de façon analogue aux autres portes, on a l'output gate. Elle filtre l'information qu'on décide de transmettre à la prochaine cellule ou couche. Puis on applique ce filtre à notre cell state qui est passé dans une $tanh$ afin de mettre ses valeurs entre -1 et 1.

$$o_t = \sigma(W_o[h_{t-1}, \mathbf{X}^t] + b_o)$$
$$h_t = o_t tanh(C_t)$$

Les différents graphiques représentant les étapes du LSTM sont disponibles en annexe A.1.

1.8 Travailler avec des données déséquilibrées

La transcription de notre problème en un problème de classification "grave vs non grave" peut être une tâche complexe. En effet, comme vu dans la partie TVE, ce qui définit ce type de sinistre, c'est à la fois son coût extrême, mais aussi une fréquence de réalisation faible. À partir de là, modéliser les évènements peut être compliqués. Il se peut que le modèle les considère comme des données aberrantes de par leur rareté dans la base. On peut citer deux grandes familles de techniques qui sont utilisées pour combattre ce problème dû au déséquilibrage des classes. L'idée générale de ces méthodes est d'obtenir une data set qui rendent la modélisation de la classe déséquilibrée un peu plus juste.

Il est à noter que, modéliser un jeu de données déséquilibré, est un sujet de recherche d'actualité toujours ouvert [16] [15] malgré la pléthore de méthodes existantes. En effet, bien que ici nous allons vous présenter les méthodes les plus connues et les plus utilisées, on dénombre sous Python 85 implémentations différentes du SMOTE [118] dont un benchmark a été réalisé [117]. De plus, on peut noter aussi la thèse de Hamdi (2012) [93] qui en plus de faire l'état de l'art sur les méthodes déjà existantes, propose une technique d'échantillonnage structurel adaptatif qui permet de rééquilibrer les données par sous-échantillonnage de la classe majoritaire [94], c'est ce qu'on appelle l'Undersampling.

1.8.1 Undersampling

L'Undersampling consiste à enlever des observations présentent dans notre classe majoritaire. Il existe trois façons de l'appliquer :

- Le Random Undersampling
- Le Cluster Undersampling
- Le Tomek links

Ces trois techniques sont détaillées dans les sous-sections ci-dessous.

1.8.1.1 Le Random Undersampling

La première, consiste à enlever de façon aléatoire des observations. C'est certainement la plus facile à mettre en place, mais elle ne prend pas en compte certains aspects du jeu de données. En effet, on peut supposer qu'il existe dans notre classe majoritaire des observations dites redondantes (i.e. qui ont exactement les mêmes caractéristiques). Il peut sembler plus judicieux par exemple d'enlever celle-là plutôt que d'autres observations de cette même classe porteuses d'une information différente.

1.8.1.2 Cluster

Le Clustering Undersampling [134] repose sur le nombre d'observations présentes dans la classe minoritaire. En effet, on crée autant de clusters de la classe majoritaire qu'on a de points dans la classe minoritaire. Une fois nos clusters créés, les centroïdes, trouvé à l'aide d'un K-means, vont donc représenter nos clusters, et donc nos observations, lors de l'apprentissage des données.

1.8.1.3 Tomek links

Le Tomek links [204] enlève les observations de la classe majoritaire qui sont proches des observations d'autres classes. On cherche, ici, à minimiser la distance entre les observations d'une même classe. On définit le Tomek Link de la façon suivante, soit deux observations x_i et x_j tel que x_i appartienne à la classe minoritaire et x_j à la classe majoritaire. On définit $d(x_i, x_j)$ la distance entre ces deux observations. Le couple (x_i, x_j) est appelé Tomek link s'il n'existe pas d'autre observation x_k telle que $d(x_i, x_k) < d(x_i, x_j)$ or $d(x_j, x_k) < d(x_i, x_j)$. C'est en utilisant la distance entre observations qu'on peut, soit enlever ce qui peut être du bruit (de par la présence d'un élément de la classe majoritaire dans l'espace de la classe minoritaire), soit enlever deux classes qui se superposent au profit de la moins présente.

Comme évoqué en introduction de cette sous partie, il existe deux grandes familles de méthodes de rééquilibrage. Comme on l'a vu, l'Undersampling consiste à diminuer le nombre d'occurrences de la classe majoritaire. Une autre possibilité de rééquilibrage consiste à augmenter le nombre d'occurrences de la classe minoritaire, c'est ce qu'on appelle l'Oversampling.

1.8.2 Oversampling

L'Oversampling consiste à créer ou dupliquer des observations présentent dans notre classe minoritaire. Comme pour l'Undersampling, il existe plusieurs façons de l'appliquer.

1.8.2.1 Random

Comme pour l'Undersampling, il existe une façon aléatoire d'appliquer l'Oversampling. Ici, on tire aléatoirement une observation qu'on duplique un certains nombres de fois. Cela revient donc à appliquer un poids à l'observation tirée. On ne crée pas une nouvelle source d'information on l'a dupliqué.

1.8.2.2 SMOTE : Synthetic Minority Over-sampling Technique

C'est certainement l'une des méthodes d'oversampling les plus utilisées pour créer de la nouvelle donnée, elle est utilisée en médecine [210] aussi bien qu'en assurance [205]. Introduite pour la première fois par Nitesh V. Chawla [39], cette méthode d'Oversampling est basée sur les plus proches voisins avec une distance Euclidienne entre les points de données dans l'espace des caractéristiques. Elle consiste à générer de nouveaux individus minoritaires, situés aléatoirement sur des segments entre individus voisins de la classe minoritaire, dans l'espace des variables explicatives.

Algorithm 2 Algorithme SMOTE

Require:

- N Nombre d'individus de la classe minoritaire
- T taux de SMOTE (<1 ou entier non–nul)
- K Nombre de plus proches voisins k.

if T<1 **then**
  Randomiser le vecteur des T individus
  $N \leftarrow N.T$
  $T \leftarrow 1$
end if
$N_f \leftarrow Nombredevariables$
Old (TN_f) : matrice des individus originaux de la classe minoritaire
New $(N.TxN_f)$: matrice des individus synthétiques générés
$Newind \leftarrow 1$: variable de comptage des individus synthétiques générés
for $i \leftarrow 1$ **to** N **do**
  Calculer les k plus proches voisins de l'individu i, dont les index correspondants (dans la matrice Old) sont stockés dans $\overrightarrow{V}(k1)$ vecteur contenant les voisins
  for $j \leftarrow 1$ **to** T **do**
    Choisir aléatoirement un nombre entre 1 et k, qui indexe l'un des k voisins dans $\overrightarrow{V}$
    for $f \leftarrow 1$ **to** N_f **do**
      $d = Old[V[j], f] - Old[i, f]$
      $ecart = U$ uniformément distrbué sur $[0, 1]$
      $New[Newind, f] = Old[i, f] + d.ecart$
    end for

    $Newind \leftarrow Newind + 1$
  end for

end for

return $N.T$ individus synthétiques de la classe minoritaire.

L'algorithme SMOTE reposant sur les k plus proches voisins, le nombre de voisins est un paramètre important à prendre en compte. En effet, si k est trop grand, on risque d'utiliser un voisin qui se trouve très loin dans notre espace, et donc générer des observations de la classe minoritaire à un endroit ou il n'y en avait pas. À l'inverse, un k trop petit, peut lui équivaloir à une méthode Random.

1.8.3 Recommandation sur l'Oversampling et l'Undersampling

En général, il est préférable d'appliquer des méthodes hybrides. Les méthodes hybrides sont une combinaison d'une méthode d'Oversampling avec une méthode d'Undersampling. Par exemple, appliquer un Tomek links (undersampling) avec un SMOTE (oversampling) [16].

De par la manipulation de la distribution de notre jeu de données, soit en créant, soit en supprimant des observations de la base originale. Cela peut introduire du biais lors de l'apprentissage. En effet, on peut par exemple être amené à appliquer un taux d'oversampling trop important et donc avoir tendance à prédire beaucoup trop de données issues de la classe minoritaire. C'est pourquoi, il peut être intéressant d'appliquer des méthodes de boostrap, comme on a pu le faire dans cette thèse, afin de réduire le biais introduit par une méthode de rééquilibrage.

1.8.4 Stratégies Algorithmiques

Pour le moment, nous n'avons présenté que des méthodes agissant sur la base, pour combattre le problème de l'asymétrie de nos classes. Il existe également des approches pour agir directement sur le modèle, afin que celui-ci prenne en compte la caractéristique de nos données.

1.8.4.1 Modification du seuil de décision

Une autre stratégie algorithmique propose de modifier le seuil de décision. En effet, plusieurs algorithmes de Machine Learning fournissent soit la classe, soit une probabilité pour chaque individu d'appartenir à une classe. Dans le cas d'une classification binaire, si la probabilité d'être dans une classe est supérieur à 50%, le modèle prédira cette classe là. Dans le cas d'une sortie multi-class, l'algorithme utilise le $\arg\max$. On prend donc une décision en fonction de ce seuil de probabilité fixé. On peut modifier la valeur de ce seuil afin de prendre en compte le problème de déséquilibrage des données, et ainsi pouvoir optimiser notre métrique.

1.8.4.2 Cost sensitive et fonction de coût

L'apprentissage sensible aux coûts [64] [201] (Cost sensitive) est une autre stratégie algorithmique. L'idée est d'agir directement sur la fonction de coût du modèle. Cela consiste à fixer des coûts inégaux sur les différents types d'erreurs de mauvaise classification. Cette approche est de plus en plus répandus, car elle rend le modèle sensible aux coûts. Metacost une approche proposée par Domingos [59] consiste à estimer pour chaque individu sa classe optimale, c'est-à-dire, celle qui minimise le coût. On peut aussi penser à la Focal Loss [133] de Facebook ou la Weighted Cross entropy [12] que nous avons utilisé, comparé et présenté un peu plus tard dans cette thèse.

Le choix d'un seuil ou de la bonne fonction de coût sont des hyper paramètres du modèle qu'il est important d'étudier. D'autres hyperparamètres jouent un rôle majeur sur les performances du modèle, c'est le cas du Batch-Size pour les réseaux de neurones.

1.8.4.3 L'influence du Batch-size

Comme vu précédemment, il existe trois types de descente de gradient le Batch, le Stochastic et le Minibatch. La taille de notre Mini-Batch a une influence sur les performances de nos modèles. Un batch de faible taille a l'avantage d'offrir un effet de régularisation et une erreur de généralisation

faible, car ils sont bruyants pour un coût de calcul donné, sur un nombre d'itérations importantes [141]. Pour les architectures profondes, Bengio suggère de prendre des batchs de taille 32 [19]. La taille du batch est un hyperparamètre du modèle à prendre en compte encore plus dans le cadre d'un de jeu de données déséquilibrées. En effet, dans le cadre du Chapitre 3, nous avons été amenés à tester différentes tailles de Batch sur un jeu de données déséquilibrées. Nous avons constaté (voir Figure 1.27) que plus le batch est petit meilleur est le F1-Score. Cela s'explique par le fait qu'avec très peu d'observations positives, le modèle passera plus de temps sur des exemples négatifs qu'à apprendre des positifs. Avec une taille de batch élevé, la pondération d'une erreur sur un négatif est beaucoup moins importante que sur un batch de petite taille.

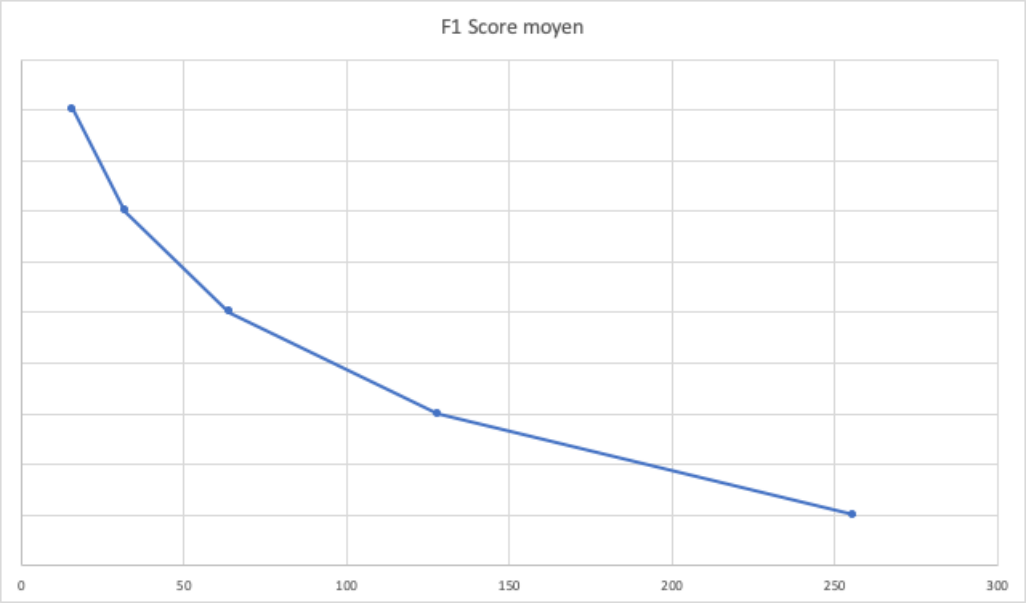


FIGURE 1.27 – Comparaison du F1 Score en fonction de la taille du Batch

1.8.5 Le choix de la métrique

Il est nécessaire de bien choisir la métrique d'évaluation. Il faut que cette métrique soit d'abord cohérente avec les objectifs, mais aussi pertinente pour répondre à la problématique. La métrique d'évaluation du modèle la plus utilisée est l'accuracy, elle indique le taux de bonnes prédictions réalisées par le modèle. Cette métrique peut être trompeuse lorsque les données sont déséquilibrées [192] [4]. Dans la suite, on note,

- TN le nombre d'observations négatives bien classifié, appelé aussi "True Negatives".
- FP le nombre d'observations négatives mal classifié, appelé aussi "False Positives".
- FN le nombre d'observations positives mal classifié, appelé aussi "False Negatives".
- TP le nombre d'observations positives bien classifié, appelé aussi "True Positives".

Nous définissions également le Rappel (Recall) et la Précision (Precision) ainsi que le F_1-score

$$\begin{aligned} \text{Recall} &= \frac{TP}{TP+FN}, \\ \text{Precision} &= \frac{TP}{TP+FP}. \\ F_1 &= \frac{2 \times Recall \times Precision}{Recall + Precision}. \end{aligned}$$

Le rappel indique la capacité du modèle à prédire correctement tous les positifs et la précision indique la capacité du modèle à ne prédire que les positifs. Ce sont des métriques pertinentes à utiliser quand on s'intéresse principalement à la classe positive. Néanmoins, elles sont asymétriques. Autrement, si on inverse la classe positive et la classe négative, elles n'auront plus la même valeur.

De même, le score F_1-score est très intéressant quand il nous permet de trouver le bon compromis entre un bon rappel et une bonne précision. Il s'agit d'une moyenne des deux métriques.

1.9 Organisation de la thèse

Les chapitres 2 à 4 de cette thèse visent à présenter la façon dont nous avons abordé le problème du provisionnement des sinistres graves dans le contexte industriel qui était le nôtre.

1.9.1 Chapitre 2

Ce chapitre fournit une présentation des données et des traitements qui ont été nécessaires à leur utilisation dans le cadre de cette thèse. Cet étape de rassemblement des données et de leur préparation a été une étape importante, notamment parce qu'elle s'inscrivait dans une volonté de Pacifica de consolider ses données en vue d'exploiter efficacement leur information. La richesse de ces données est essentielle dans l'optique des objectifs d'analyse des sinistres graves, développée par la suite.

1.9.2 Chapitre 3

Dans ce chapitre, nous développons une méthodologie pour classer automatiquement les rapports des sinistres. Ces rapports sont établis à l'ouverture du sinistre. À partir de cette analyse automatique, le but est de prédire si un sinistre sera grave ou non. La difficulté provient du déséquilibrage de la distribution des deux classes, trop peu d'observations sont associées à une étiquette positive. De par la nature des données (non structuré), les modèles classiques tels que la régression logistique ou les arbres ont quelques difficultés à bien labelliser les observations. Les données étant déséquilibrées, nous proposons différents algorithmes de rééquilibrage pour faire face à ce problème. Nous discutons de l'utilisation des différentes méthodologies d'intégration utilisées pour traiter les données textuelles et du rôle des réseaux de neurones. Ce chapitre reprend le contenu d'un article publié dans Annals of Actuarial Science.

1.9.3 Chapitre 4

Dans cet article, nous développons une méthodologie afin de prédire le coût d'un sinistre d'assurance ouvert (RBNS, Reported But Not Settled) à partir d'un ensemble de covariables complexes

avec diverses structures de données (données structurées et non structurées). La technique combine différentes architectures de réseaux de neurones profonds (telles que les LSTM pour les données textuelles) avec des méthodes de prédiction d'analyse de survie (pour prédire le moment du règlement de la réclamation). Les méthodes de Deep Learning sont utilisées pour extraire des caractéristiques de nos données complexes afin d'effectuer une réduction de dimension. Ces caractéristiques peuvent être connectées à un réseau de neurones "final" ou combinées à des modèles plus intelligibles comme un modèle linéaire généralisé, si le besoin d'interprétation est plus important que la qualité de la prédiction. Une analyse de données réelles illustre la technique.

1.9.4 Chapitre 5

Enfin, dans un dernier chapitre, nous présentons une conclusion générale ainsi que les perspectives envisagées pour la poursuite de ce travail.

Chapitre 2

Présentation des données et premiers traitements

Ce second chapitre a pour but de présenter les différents types de données que nous avons utilisées tout au long de cette thèse ainsi que les prétraitements effectués sur celles-ci.

Rappelons tout d'abord le processus de collecte des données. Suite à la déclaration d'un sinistre par l'assuré, le gestionnaire sinistre ouvre un dossier dédié dans lequel il commence à renseigner un certain nombre d'informations. On y trouve des caractéristiques intrinsèques au sinistre ou à l'assuré, mais aussi un champ libre que le gestionnaire sinistre remplit afin de décrire les circonstances du sinistre. Par la suite, le gestionnaire met à jour certaines variables qui connaissent des variations au fil de la durée de vie du sinistre, dues à l'évolution de l'état de santé de l'assuré ou à la gestion du sinistre. L'assureur complète souvent sa base avec des données dites exogènes, qui ne sont pas directement liées à la sinistralité, mais qui peuvent intervenir afin de permettre une meilleure compréhension du risque.

Ainsi, l'assureur a à sa disposition une forte quantité de données. Ces données sont de différents types : on trouve des données structurées, c'est-à-dire des données ayant une typologie, un format et une longueur définis, comme les informations intrinsèques aux sinistres présentes dans les bases de données ou les données externes, on a également des données temporelles, qui sont des données qui évoluent au fil du temps telles que les variables liées au développement des sinistres, mais aussi des données non structurées comme les commentaires du gestionnaire. Leurs différents types impliquent un traitement des données différent, ce que nous vous présenterons par la suite.

Enfin, il nous semble important de revenir sur la Théorie des Valeurs Extrêmes, et notamment sur la détermination du seuil de gravité, car elle joue un rôle primordial dans la compréhension et la modélisation de notre charge.

2.1 Les données

Le Big Data est souvent associé à la règle dite des "3V". Le premier pour "Volume", les algorithmes de Data Science sont conçus pour traiter une quantité de données considérable. Le second "V" correspond au terme "Variété". En effet, les modèles de Machine Learning permettent l'utilisation d'une grande variété d'informations de différents types (structurées ou non structurées) et venant de diverses sources (données internes, Open Data). Enfin, le dernier "V" représente la "Vélocité", autrement dit la fréquence de création, de collecte et de partage de ces données. Dans cette sous-partie,

nous vous présenterons les données que nous avons utilisées durant cette thèse. Nous nous concentrerons principalement sur la vélocité des données disponibles tout d'abord en interne, à partir des bases de données de l'assureur, puis nous nous pencherons sur les données Open Source dans un second temps.

L'assureur dispose de plusieurs types de données. Certaines sont issues des informations contrat, d'autres sont liées aux sinistres et sont parfois vouées à évoluer au fil du temps. Il y a également des données qui ont vu leur utilisation évoluer grâce à l'émergence des réseaux de neurones.

Caractéristiques intrinsèques

Certaines informations issues du contrat ont un rôle important dans la prédiction de la charge. On pense notamment à l'âge de la victime, sa zone géographique, son type de véhicule, sa profession. Bien que ces variables ne portent aucune information directement liée au sinistre, elles n'en sont pas moins importantes. Prenons comme exemple l'âge de la victime, celle-ci aura plusieurs implications directes sur la durée de vie du sinistre et sa liquidation. Si la victime est mineure, l'assureur doit attendre sa majorité afin de bien évaluer les conséquences sur son état physique. Cet aspect va donc jouer sur la durée de vie du sinistre. Si la victime est un jeune actif dont l'accident implique une perte de gains professionnels, l'assureur va devoir la prendre en charge pendant une période plus importante que dans le cas où la victime est une personne proche de la retraite. Par contre, une personne âgée se remettra moins bien d'un accident qu'une personne plus jeune.

D'autres variables sont quant à elles directement liées au sinistre, comme la garantie souscrite pour couvrir le sinistre, ou bien le type de sinistre (matériel et/ou corporel). La particularité de ces variables est qu'elles sont statiques et ne sont pas vouées à évoluer au cours de la vie du sinistre contrairement aux variables de développement.

Variables de développement

Au cours de la vie du sinistre, plusieurs facteurs impactant directement la charge peuvent évoluer.

Tout d'abord, l'état physique de la victime étant rarement stable, plusieurs postes le décrivant sont amenés à évoluer. On pense notamment aux postes suivants :

- Taux de Déficit Fonctionnel Permanent
- Pertes de Gains Professionnels Actuels
- Pertes de Gains Professionnels Futurs
- Dépenses de Santé
- Souffrance Endurée
- Préjudice Esthétique Permanent
- Assistance Tierce Personne

Ces postes représentent à eux seuls plus de 80% de la charge totale corporelle, la moindre variation de l'un d'entre eux aura donc un impact direct sur la prédiction.

D'autres variables décrivant la gestion du sinistre sont aussi susceptibles d'évoluer. La première concerne les recours, ils peuvent être rattachés aux paiements (on parle alors de recours encaissés) ou aux provisions dossiers/dossiers (on parle alors des prévisions de recours). Afin de ne pas fausser notre étude, nous avons décidé de travailler sur des charges brutes de recours. En effet, si nous regardons les charges nettes de recours, deux sinistres ayant les mêmes caractéristiques peuvent avoir des charges différentes en raison des recours. On aurait donc intégré un biais dans notre étude.

Enfin un aspect non moins important est le mode de transaction. Bien que dans la majorité des cas, le règlement du sinistre s'effectue à l'amiable entre l'assureur et l'assuré, il arrive qu'un petit pourcentage des sinistres finissent au tribunal. Un tel mode de règlement implique des démarches qui peuvent être longues et coûteuses. Bien qu'ayant une faible représentation, un seul de ces sinistres peut impacter à lui seul, et de façon importante, la charge sur un exercice.

La particularité de ces données réside dans leur mise en forme. Afin de pourvoir les traiter avec un modèle type LSTM, les données sont stockées sous la forme d'une matrice à trois dimensions (nombre d'observations ; pas de temps ; variables).

Commentaires des gestionnaires sinistres

Nous allons aborder ici un type de données dont l'utilisation commence à éclore dans le monde de l'assurance. Ce sont des données qui, à la base, n'étaient pas destinées à la modélisation et qui ont vu leur utilisation se développer avec l'émergence des réseaux de neurones. Bien que leur collecte ne soit pas systématiquement effectuée par les compagnies d'assurance, ces données sont de plus en plus exploitées et constituent une source d'information intéressante dans le cadre d'analyses fines de la sinistralité.

Le premier type d'information non structurée qui peut être appréhendé dans le cadre d'une compagnie d'assurance concerne les commentaires renseignés par les gestionnaires sinistres. Ces commentaires sont relatifs à l'enregistrement d'un sinistre donné et sont stockés directement sous forme de texte. Cela implique donc que, pour un sinistre ayant plusieurs victimes, chaque victime se verra attribuer le même commentaire. S'ils sont globalement de nature descriptive, la plupart des commentaires émettent un avis critique quant aux causes de l'accident considéré. En effet, nous retiendrons que l'un des aspects métier des gestionnaires consiste à évaluer de façon précise les sommes d'argent mises en jeu en conséquence de la survenance d'un sinistre au titre des engagements qui ont été pris par l'assureur.

Prenons comme exemple des phrases représentant des sinistres non graves et des sinistres graves.

— sinistre non grave :
  1. "Carambolage sur la route, 58 véhicules impliqués."
  2. "L'assuré se fait heurter à l'arrière du véhicule, choc au pare-choc arrière."

— sinistre grave :
  3. "L'assuré heurte le sol d'un rond-point et meurt lors du transfert en urgence."
  4. "Tiers ne s'arrête pas au panneau stop et percute l'assuré sur le côté avant droit, l'airbag se déclenche."

Comme nous pouvons le voir, la qualité et le détail des renseignements saisis sont manifestes, mais témoignent des difficultés relatives à l'analyse de telles informations. Ce constat est notamment dû au fait que la plupart des données de commentaires sont écrites par plusieurs humains et n'étaient pas destinées à être communiquées ou analysées. Cela implique qu'elles ne sont que très peu standardisées, et c'est un premier problème qui en engendre d'autres. Le deuxième est la subjectivité que pourrait apporter le gestionnaire lors de la rédaction du compte rendu, celle-ci pourrait fausser le jugement d'un modèle. Enfin, le champ texte étant un champ libre, le gestionnaire peut utiliser des abréviations ou encore faire des fautes de frappe.

Comme nous avons pu le constater, les données internes de l'assureur sont très variées. On y distingue des données structurées et des données non structurées, des données temporelles et des données "fixes". À ces données internes, nous avons décidé d'ajouter des données Open Source.

2.1.1 Données Open Source

En effet, l'ouverture de données publiques en France dans le cadre global de l'Open Data est une véritable opportunité pour de nombreux acteurs de l'économie, notamment ceux évoluant dans le secteur de l'assurance. Beaucoup de données sont disponibles et apportent plusieurs informations à des mailles géographiques différentes et sur des facteurs sociaux-environnementaux et comportementaux différents. Les données INSEE sont issues du site internet de l'INSEE. Elles sont triées par catégories : logement, socio-économiques, socio-démographiques et points d'intérêt. Une telle source d'information nous permet d'identifier par exemple le salaire moyen d'un département ou d'une ville, information d'une réelle importance puisque dans la prise en charge des pertes de gains professionnels, à caractéristiques équivalentes, une variation de salaire implique une variation de la charge.

D'autres sources de données peuvent nous apporter des informations sur la sinistralité au niveau départemental.

Association pour la Gestion des Informations sur le Risque en Assurance : AGIRA

L'AGIRA a été créée par la Fédération Française de l'Assurance (FFA), elle regroupe les sociétés d'assurance exerçant sur le marché français et les organisations professionnelles intervenant dans le secteur assurantiel.

L'AGIRA tient un Fichier des Victimes Indemnisées (FVI) qui a pour objet d'informer le public des indemnités allouées aux victimes d'accidents de la circulation dans le cadre de décisions prises soit par accord transactionnel, soit par voie judiciaire. Ce fichier s'inscrit dans le cadre de l'article 26 de la loi du 5 juillet 1985 tendant à l'amélioration de la situation des victimes d'accidents de la circulation et à l'accélération des procédures d'indemnisation. Ce fichier a pour but de connaître, à sinistre équivalent, les indemnités versées aux personnes blessées conservant une incapacité ou aux victimes indirectes (ayants droit ou proches) de personnes décédées.

Un tel fichier nous permet d'établir une classification des départements ayant une plus forte sinistralité, non plus liée à la spécificité de l'unique portefeuille de l'assureur, mais à une vision plus globale du marché.

Les données AGIRA nous permettent ainsi d'identifier les départements les plus sinistrés tandis que les données BAAC nous permettent d'avoir une meilleure vision du comportement routier.

Bulletins d'Analyse des Accidents Corporels : Fichiers BAAC

Les accidents corporels ont été définis dans l'arrêté du 27 mars 2007 relatif aux conditions d'élaboration des statistiques. L'ONISR, en charge de l'administration et de la diffusion des statistiques d'accidentalité aux termes du décret du 15 mai 1975 relatif au CISR, a précisé les modalités de prise en compte des accidents.

"Est classé comme accident corporel de la circulation tout accident, avec une victime, impliquant au moins un véhicule sur une voie ouverte à la circulation publique ; quel que soit l'événement causal, hors acte intentionnel de type suicide ou homicide.

Les fichiers BAAC sont remplis par les forces de l'ordre à la suite de tout accident corporel où ils sont appelés. Le fichier est complété par les observatoires départementaux de sécurité routière. Les

données brutes ayant servi au bilan font l'objet d'un recueil mis également en ligne. Avoir accès à ces données est une opportunité pour l'assureur. Ainsi, il peut ainsi dresser un portrait comportemental par département en répondant à ce type de questions :

Est-ce un département où les individus mettent souvent la ceinture en voiture ou le casque en deux roues ?

La chaussée est-elle éclairée ?

La chaussée est-elle glissante ?

Ces questions permettent de comprendre si c'est à cause de la route qui est dangereuse ou si c'est à cause du comportement humain qu'un département est plus à risque.

Bien que la majorité de nos données soient déjà sous forme numérique, certaines doivent être traitées afin d'être rendues "lisible" par nos modèles.

2.2 Traitement des données structurées

En Machine Learning, le prétraitement des données est l'étape au cours de laquelle les données sont transformées ou codées, pour les amener à un état tel que le modèle peut facilement les analyser. C'est donc une étape importante et nécessaire pour le bon apprentissage du modèle.

Il existe deux grandes familles de données structurées qui sont elles-mêmes divisibles en deux sous-familles :

Les données qualitatives (ou catégorielles) : Ce sont des variables prenant un nombre fini de valeurs. Les valeurs qu'elles prennent sont appelées des catégories ou modalités. Ces dernières sont exprimées sous forme littérale. Parmi ces données, on distingue les variables nominales des variables ordinales. Les variables nominales sont des variables n'impliquant aucun ordre, comme la couleur (on ne peut pas dire que rouge est supérieur à blanc). À l'inverse les variables ordinales sont des variables catégorielles qui ont une notion d'ordre (un senior est plus âgé qu'un enfant par exemple).

Les données quantitatives (ou numériques) : Ce sont les variables qui prennent des valeurs numériques, à condition que ces valeurs expriment une quantité et aient un sens lorsque l'on y applique des opérations arithmétiques. Une variable quantitative est soit discrète, soit continue. Si le nombre de valeurs possibles (et probables) d'une variable est très grand, alors on peut la considérer comme continue (salaire, durée entre deux dates, etc). Sinon, on la considère comme étant discrète (nombre d'enfants, nombre de pièces, etc).

Dans cette sous partie, nous présenterons les différentes étapes de prétraitement (ou preprocessing en anglais) que nous avons appliquées sur nos données. Avant de rentrer dans les spécificités de chacune, nous commencerons par vous parler d'une technique qui peut s'appliquer aux deux familles, l'agrégation.

L'agrégation

L'agrégation des données est effectuée afin de mettre dans une meilleure perspective l'information portée par les données.

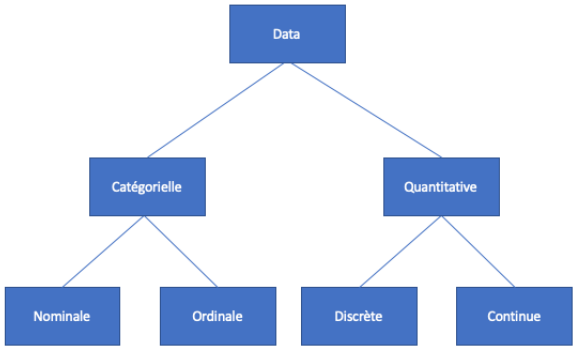


FIGURE 2.1 – Les différents types de variables

L'agrégation d'une variable numérique nous aide à réduire des dizaines, voire des centaines de valeurs, diminuant ainsi la dimension de celle-ci. L'agrégation d'une variable qualitative nous permet de regrouper sous forme de classes des individus portant un risque similaire.

Cette technique implique une réduction de la consommation de mémoire et du temps de traitement, ce qui est un avantage considérable. L'agrégation nous fournit une vue de haut des données car le comportement des groupes ou des agrégats est plus stable que celui des données individuelles. Le fait d'agréger une variable s'appelle la discrétisation. Cette technique peut s'appliquer tant aux données de types quantitatives (regroupement d'individus par classe d'âge par exemple) qu'aux données de types qualitatives (regroupement de modèles de véhicules par types de véhicules)

Bien qu'elle s'applique aux deux familles de variables présentées précédemment, chaque famille possède aussi son type de traitement.

2.2.1 Traitement des données quantitatives

Ce type de variables peut certes déjà être utilisé de manière brute pour la modélisation. Néanmoins, il existe une famille de prétraitement appelé le Scaling qui a pour but d'optimiser la vitesse de convergence du modèle ainsi que ses performances.

Scaling et Standardisation

La mise à l'échelle (ou scaling en anglais) est importante si deux variables quantitatives ont des échelles de valeurs différentes. Le scaling nous permet de nous assurer qu'une variable ne prendra pas un poids plus important dans le modèle à cause de son échelle de représentation. Par exemple, si on utilise l'âge et le salaire d'une personne en variables d'entrée du modèle, certains algorithmes feront, à tort, davantage attention au salaire simplement parce qu'il est plus grand.

La normalisation est une forme de scaling. Elle implique la transformation des données en une distribution normale. Les algorithmes utilisant une descente de gradient convergent beaucoup plus rapidement sur des données normalisées, il est donc logique d'appliquer ce traitement aux données pour obtenir de meilleurs résultats.

Ce type de traitement est spécifique aux données numériques, pour les données catégorielles nous utilisons d'autres techniques.

$$X_{standardisé} = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad X_{normalisé} = \frac{X - \mu}{\sigma}$$

FIGURE 2.2 – À gauche une forme de scaling où la variable sera comprise entre 0 et 1. À droite la formule de la standardisation ou μ est la moyenne de la variable et σ sa variance

2.2.2 Traitement des données qualitatives

Les modèles de Machine Learning prennent en entrée des variables numériques. Dans le cas des données qualitatives, celles-ci sont souvent sous forme de texte comme la marque de la voiture, ou bien les valeurs d'une classe d'âge (enfant, senior, actif...). Il faut donc les traiter de façon à les rendre "lisibles" par le modèle. Comme nous l'avons dit précédemment, il existe deux types de variables catégorielles différents, induisant ainsi un prétraitement différent.

Label Encoder

Commençons par les variables ordinales. Étant donné que ces variables ont un ordre, il est important de le garder afin que le modèle l'apprenne. Pour cela, on peut "labelliser" les variables. Cette approche consiste simplement à convertir chaque valeur d'une colonne en un nombre, en veillant à garder l'ordre de la variable.

Le Label Encoder est très utile pour les variables ayant une relation d'ordre naturel, pour les autres types de variables catégorielles, nous leur préférons le One-Hot Encoding.

One-Hot Encoding

Pour les variables catégorielles où une telle relation ordinale n'existe pas, l'encodage en nombre entier peut entraîner de mauvaises performances ou des résultats inattendus. Dans ce cas, un codage one-hot peut être appliqué à la représentation entière, cette variable deviendra une matrice. Si la variable a K modalités, on l'encode comme une matrice $\mathbf{X} = (x_{a,b})_{1 \leq a \leq m, 1 \leq b \leq K}$, avec m lignes et K colonnes, la $i-$ème colonne représente la $i-$ème modalité de la variable, notée $\mathbf{x}_i$. Si pour l'observation j la variable vaut cette modalité, $x_{j,k} = 1$ sinon $x_{j,k} = 0$. À noter que $\sum_{b=1}^{K} x_{j,b} = 1$, c'est-à-dire que qu'on ne peut avoir qu'un seul élément de la ligne valant 1.

2.3 Traitement du texte

Le texte brut se prête rarement directement à l'analyse. Il faut le traiter pour le rendre "lisible" par nos modèles. C'est le but du prétraitement des données textuelles. Il s'agira autant de simplifier le texte, de le nettoyer, de le réduire, que de l'enrichir en annotant ses termes d'attributs particuliers relatifs à leur nature lexicale ou syntaxique. La thèse de Balech (2019) [13] fait l'état de l'art de

toutes les techniques disponibles pour traiter le texte. Nous présenterons ici seulement les étapes de prétraitement que nous avons utilisées.

2.3.1 Le nettoyage du texte

L'une des premières étapes consiste en une suite d'opérations élémentaires de normalisation du texte qui réduisent le nombre de formes, mais peuvent aussi détruire de l'information :

- Mettre le texte en minuscule.
- Supprimer la ponctuation.
- Éliminer les nombres et dates.
- Éliminer aussi les mots sans signification tels que les pronoms, on appelle cela des "stopwords".
- Corriger l'orthographe.
- Supprimer les mots peu ou trop présents.

Une seconde étape du pre processing du texte est la lemmatisation ou le *stemming*. Le processus de « lemmatisation » consiste à représenter les mots sous leur forme canonique. Par exemple pour un verbe, ce sera son infinitif ; pour un nom, son masculin singulier. L'idée étant de ne conserver que le sens des mots utilisés dans le corpus. Le «*stemming*» (ou racinisation en français) consiste, quant à lui, à ne conserver que la racine des mots étudiés. L'idée étant de supprimer les suffixes, préfixes et autres des mots afin de ne conserver que leur origine. C'est un procédé plus simple que la lemmatisation et plus rapide à effectuer puisqu'on tronque les mots essentiellement contrairement à la lemmatisation qui nécessite d'utiliser un dictionnaire.

Une fois cette première phase de nettoyage du texte finie, nous pouvons passer à la seconde phase, la «*Tokenisation*». Elle consiste à découper les mots des différents documents qui constituent notre corpus, autrement dit, un token est un mot. Si étudier les mots est une idée assez intuitive en NLP, l'étude des paires (2-grams), des triplets (3-grams) de mots l'est aussi. Les n-grams sont ainsi des suites de 1,2, … n mots nous permettant de saisir le contexte de nos mots.

L'étape de pre processing permet donc de nettoyer le texte, de créer de l'information. On peut facilement comparer cette étape à ce qu'on applique aux données structurées, c'est à dire la normalisation, le feature engineering. Bien que semblable, il est à noter une différence importante : une fois le prétraitement fini, la donnée structurée peut être directement lue par le modèle ce qui n'est pas le cas du texte. Nous devons donc trouver une représentation convenable du texte afin qu'il soit traité efficacement par nos modèles.

2.3.2 Label encoder et One-hot encoding

Une première idée serait de considérer les mots comme une variable catégorielle. Comme on l'a vu précédemment, on pourrait soit créer un dictionnaire qui associe à un mot un nombre, soit utiliser le one-hot encodding pour encoder nos mots en vecteur. Bien que ces deux méthodes permettent de rendre lisible nos données par nos modèles, elles ont deux inconvénients non négligeables. Le premier concerne la taille du vocabulaire. Soit V la taille de notre vocabulaire, V peut être très grand, cela pose un problème pour l'utilisation du One-Hot. En effet, la matrice issue de notre traitement serait une matrice très *sparse*. Le deuxième problème concerne la notion de distance entre nos mots. L'utilisation du Label Encoder crée une notion d'importance entre les mots qui n'existe peut-être pas dans la réalité. Quant au One hot, par définition, tous les mots ont la même distance.

Nous ne pouvons pas utiliser ces deux méthodes à cause des problèmes évoqués. En effet, il est important de savoir identifier des synonymes ou des mots proches entre eux. Pour cela, nous devons trouver un moyen de représenter nos mots dans un espace où il n'y pas de perte d'information.

2.3.3 Word Embedding

L'idée du *Word Embedding* est d'associer à chaque vecteur "one hot encodé" un vecteur de plus petite dimension composé uniquement de nombres réels et non plus de 0 ou de 1.

L'Embedding consiste à définir une représentation dense de nos mots et elle a deux objectifs :

- Réduction de la dimension : Elle associe chaque mot à un vecteur dense de taille N ou $N < V$
- Similarité entre les mots : Les mots qui sont similaires entre eux ou très souvent associés seront proches en terme de vecteur embedded.

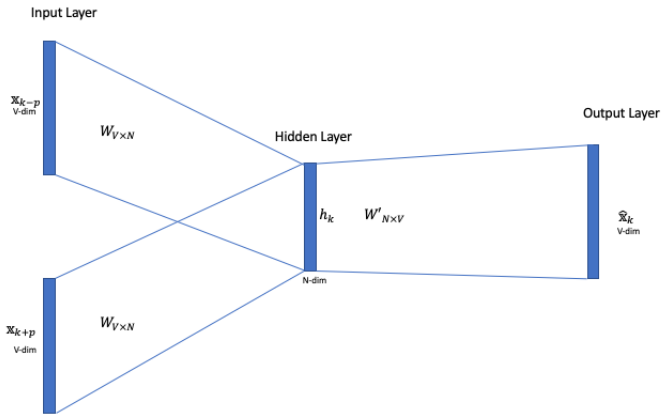


FIGURE 2.3 – Représentation de la construction de la matrice d'Embedding selon Mikolov CBOW

Pour cette tâche, d'un point de vue technique, au moins deux approches sont disponibles :

- Word2Vec proposé par Mikolov (2013)[147] de Google. Il propose de prendre en compte explicitement le contexte dans lequel apparaît un mot, c'est le modèle Continuous Bag-of-Words (CBOW) qui est le modèle opposé à Skip-Gram pour lequel à partir d'un mot on cherche à prédire les mots du contexte. Le modèle repose sur un réseau de neurones à une couche cachée (avec N, le nombre de dimensions, inférieur à V, le nombre de termes).
- Glove proposé par Pennington (2014) [161] qui présente l'avantage de construire deux vecteurs : un spécifique et un de contexte. Contrairement au Word2vec, il n'a pas recours à un réseaux de neurones.

Word2vec apporte deux innovations importante dans le traitement et la représentation du texte :

- La première est que la similarité entre deux mots dans un corpus peut non seulement être mesurée par une corrélation, mais surtout être calculée non sur l'ensemble du corpus des co-occurrences, mais sur des fenêtres glissantes de texte, de l'ordre de 5 à 10 mots consécutifs. Les mots sont considérés comme similaires car ils apparaissent dans les mêmes séquences.
- La seconde est que l'on peut utiliser le *cosinus similarity* afin d'évaluer la corrélation entre deux mots. Si le cosinus vaut -1 les mots sont opposés, s'il est proche de 1, ils expriment la même idée. L'idée clé est de représenter cette matrice de corrélations dans un espace vectoriel de grande dimension (de 50 à 1000). Contrairement à l'ACP qui cherche à réduire la dimension, l'Embedding cherche une représentation vectorielle étendue afin de conserver l'information.

Word2Vec est un modèle basé sur des fenêtres de contexte, il ne bénéficie donc pas des informations de l'ensemble du document. De plus, il ne capture pas les informations de sous-mots, ce qui pourrait être intéressant puisque les adjectifs dérivés de noms ou de verbes détiennent des informations en commun. Enfin, Word2Vec ne peut pas gérer les mots hors vocabulaire (OOV), car les mots non vus pendant la formation ne peuvent pas être vectorisés. C'est pourquoi nous avons préféré utiliser Fasttext qui est une méthode d'Embedding reposant en grande partie sur Word2Vec mais qui corrige ces problèmes.

Fasttext[27] a été développé par Facebook. C'est une méthode d'Embedding qui s'appuie sur les modèles Skip-Gram ou CBOW de Word2Vec et améliore son efficacité et ses performances, comme expliqué par les points suivants :

- Il est plus rapide et plus simple à entraîner. Sur l'évaluation de la similitude, FastText donne de meilleurs résultats que Word2Vec sur un ensemble d'apprentissage plus petit.
- Il bénéficie d'informations sur les sous-mots, FastText utilise tous les n-grams de ce mot pour calculer le score du mot.
- Il peut générer un vecteur d'Embedding d'un mot hors du vocabulaire. Un vecteur de mot OOV est construit avec la représentation vectorielle moyenne de ses n-grammes.

Cette section nous a permis d'introduire comment traiter les variables en entrée du modèle. Le problème que nous étudions est un problème de régression, c'est-à-dire que la variable de sortie est une variable continue. La loi de la variable Y est un mélange de deux lois. Afin d'identifier le seuil de séparation de ces deux lois, il est nécessaire d'avoir recours à la Théorie des Valeurs Extrêmes. La TVE peut-être utilisée pour pré-traiter la variable de sortie afin de passer d'un problème de régression à un problème de classification.

2.4 Théorie des valeurs extrêmes et détermination du seuil des sinistres graves

Dans cette section, nous souhaitons présenter l'utilisation de la Théorie des valeurs extrêmes. Bien que jusqu'à présent nous traitions des données et de leurs transformations en classe, de leur pré-traitement en variable binaire ou bien de leur représentation dans un espace continue. Il reste une variable que nous n'avons pas abordé et il s'agit ni plus ni moins de la variable à prédire, la charge du sinistre. Dans les pages précédents et suivantes, il nous est arrivé et il nous arrivera de présenter cette variable continue comme une variable binaire ayant pour modalité "grave" et "non grave". En pratique, on utilise un seuil afin de séparer une variable continue en deux. Ce seuil, qui nous permet de rendre notre variable continue en une variable de classification, est trouvé en utilisant les outils de TVE présentés au chapitre précédent.

Pour la suite de notre étude, nous avons souhaité tester la stabilité dans le temps. Pour cela nous avons découpé notre période d'étude en 3 sous-périodes de 5 années. Ce choix a été effectué en fonction du rapport Dintilhac [58] mis en place en 2006. Suite à celui-ci, nous avons constaté une forte inflation aux postes de préjudice corporel. Nous avons donc une période pré-Dintilhac (2002-2006) et deux périodes post-Dintilhac (2007-2011 et 2012-2016).

Avant de présenter les résultats de l'analyse, il nous semble important d'en démontrer le besoin. En effet, il existe un grand nombre d'outils graphiques et statistiques qui permettent d'étudier une distribution. Nous pouvons néanmoins nous demander si ils sont fiables dans notre cas.

Cette section utilise les notations de la section 1.3.

La distance relative

Algébriquement, on utilise la distance relative à une unité du centre de la distribution. Soient

- m le paramètre de tendance centrale.
- s le paramètre d'échelle.

Définition 18. *La distance relative d'une unité au centre d_i est définie par la formule suivante*

$$d_i = \frac{|x_i - m|}{s}$$

Le couple (m, s) peut prendre plusieurs formes :

- moyenne, variance
- médiane, MAD [1]
- médiane, intervalle inter-quartile

En utilisant les couples (moyenne, variance) et (médiane, MAD), on trouve les seuils suivants :

Années	Moyenne/Variance	Mediane/MAD
2002-2016	7524,17	3165
2002-2006	4140	1698,64
2007-2011	7155,09	3051,08
2012-2016	9648,08	4028,36

TABLE 2.1 – Seuils avec la distance relative

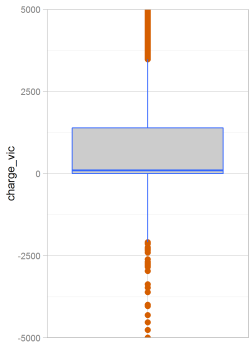
On remarque l'impact du rapport Dintilhac sur le coût de nos sinistres avec un seuil qui double presque entre la période pré et post-Dintilhac. Nous verrons plus tard comment mettre en as-if nos sinistres afin de régulariser l'inflation qui peut être liée à la fois à un contexte économique mais aussi à une gestion des sinistres différente.

Le Boxplot

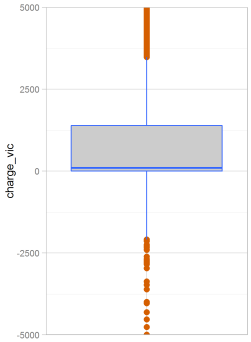
Le Boxplot est un outil graphique qui permet de détecter les valeurs aberrantes.

1. Median Absolute Deviation définie par

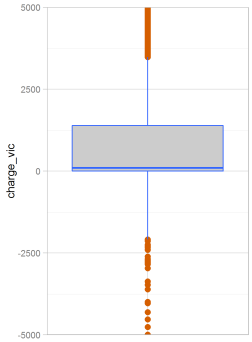
$$median_i\{|x_i - median_j(x_j)|\}$$



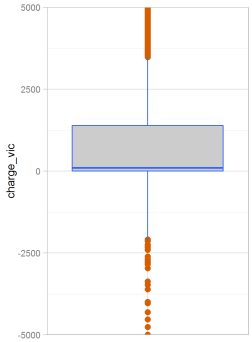
(a) 2002-2016



(b) 2002-2006



(c) 2007-2011

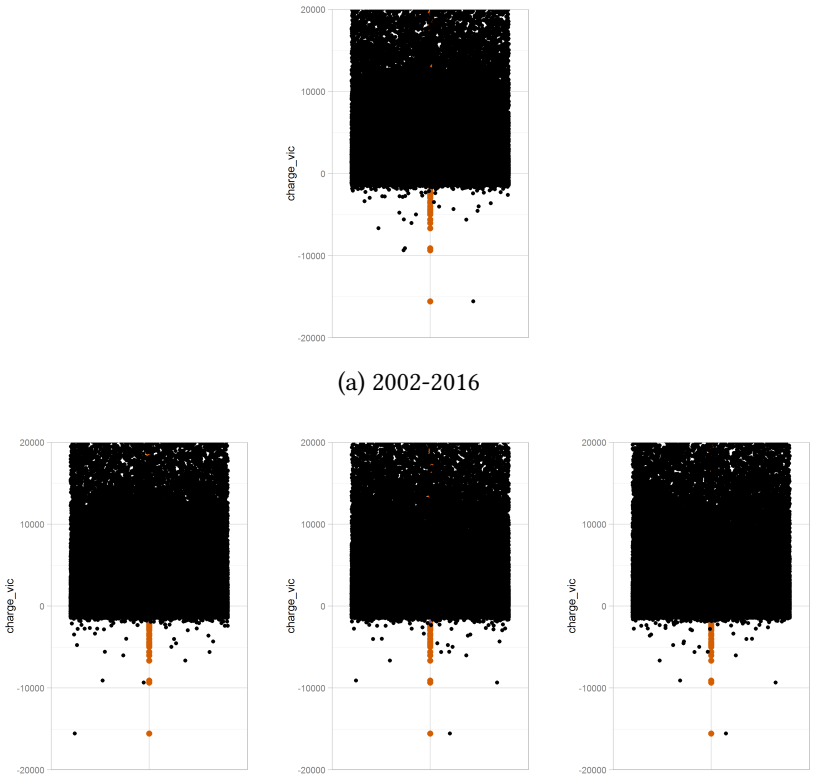


(d) 2012-2016

FIGURE 2.4 – Boxplot

Contrairement à la méthode algébrique, on a un seuil qui semble stable aux alentours des 3000€.

Mais en rajoutant la distribution sur le boxplot, on se rend compte qu'il y a un trop grand nombre de valeurs qui sont considérées comme extrêmes.



(a) 2002-2016

(b) 2002-2006 (c) 2007-2011 (d) 2012-2016

FIGURE 2.5 – Boxplot avec la distribution

Nous avons décidé de commencer notre étude par ces indicateurs afin de montrer l'importance de la Théorie des valeurs extrêmes. En effet, bien que le boxplot et la distance relative soient très utilisés pour faire de l'analyse descriptive, ils se trouvent très limités en dehors d'une distribution Normale. Nous ne pouvons donc pas nous fier à ces outils dans l'optique de séparer nos sinistres extrêmes des non-extrêmes.

Bien que le boxplot ou la distance relative ne suffisent pas à déterminer un seuil, il est intéressant de regarder la répartition du portefeuille sans ces *outliers*.

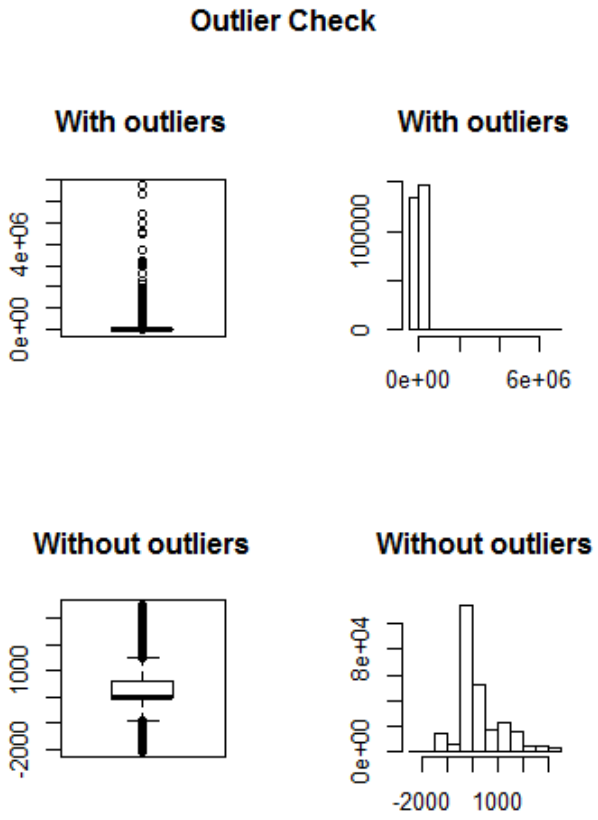


FIGURE 2.6 – Boxplot et histogramme avec et sans valeurs aberrantes

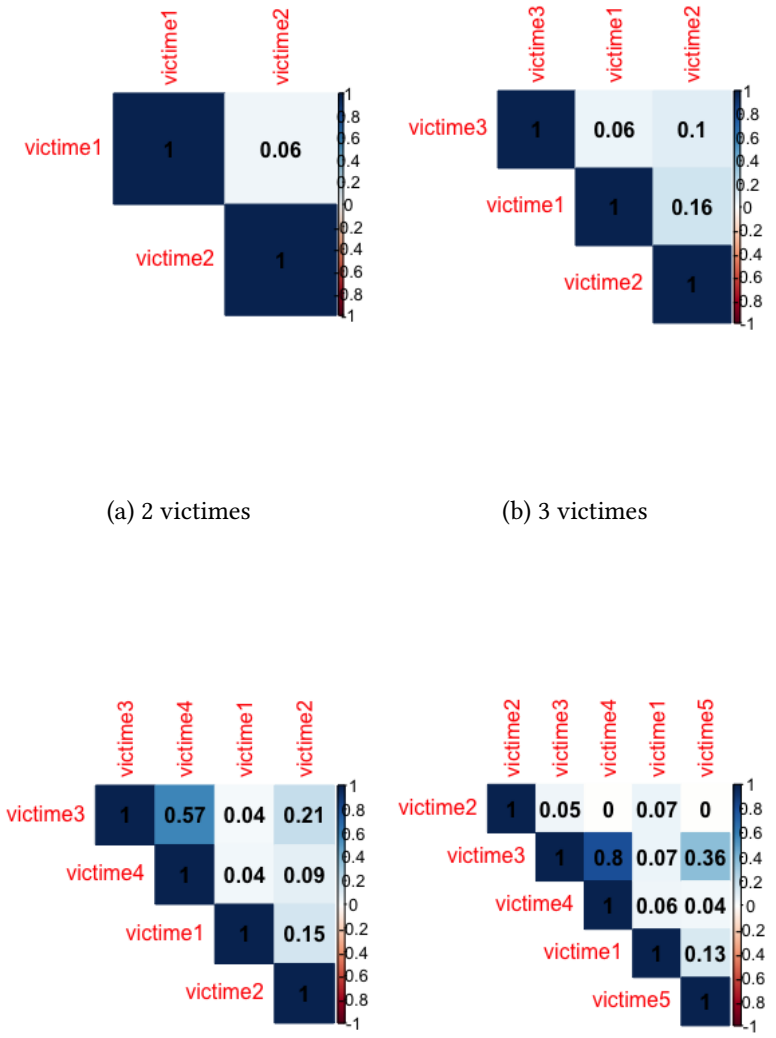
La Figure 2.6 nous montre la troncature qu'on effectuerait en se fiant aux boxplots. Une valeur extrême n'étant pas une valeur aberrante, l'utilisation des outils d'analyse de données classiques ne peut être prise en compte. De plus, l'utilisation du TCL lui n'est pas valable si $\mathbb{E}(X^2)$ ou $\mathbb{E}(X)$ n'est pas fini. Quant à la f.d.r empirque, elle fausserait nos résultats. En effet, si on cherche la probabilité qu'un sinistre ait une charge plus importante que le maximum de notre portefeuille est nulle. Il nous est donc nécessaire d'utiliser les outils de TVE.

Avant d'aller plus loin, il est légitime de se poser la question sur l'indépendance de nos observations. En effet, il arrive que pour un même sinistre nous ayons plusieurs victimes. On peut supposer l'hypothèse d'indépendance respectée car toutes les victimes n'ont pas réellement le même comportement. Une victime peut être légèrement blessée tandis que l'autre peut être décédée par exemple, dans le cas d'un carambolage l'une sera au milieu et une autre à une extrémité. Bien que intuitivement nous pouvons le supposer, nous avons tout de même effectué un test de corrélation.

Nous avons majoritairement des sinistres avec une seule victime. L'étude de la corrélation s'effectuera donc sur 20% de la base comme le montrent les chiffres du Tableau 2.2.

Nombre de victimes	Représentation dans la base
1	78%
2	16%
3	4%
4	1,5%
5	0,5%
6	0,2%
7 et plus	0,15%

TABLE 2.2 – Répartition du nombre de victimes par sinistre



(a) 2 victimes

(b) 3 victimes

(c) 4 victimes

(d) 5 victimes

FIGURE 2.7 – Matrice de corrélation des montants.

Bien que la corrélation augmente avec notre nombre de victimes, il est à noter que les résultats du test perdent en puissance de part le faible nombre d'observations. De plus, cette corrélation est due au fait que plus on a de victimes sur un sinistre plus il y a de chances qu'au moins deux aient une charge nulle, car ce type de sinistres représente 30% de notre base. La Figure 2.7 nous permet donc de conclure qu'il y a indépendance entre nos observations.

L'étude s'effectuera dans un premier temps sur les sinistres clos car la charge d'un sinistre en cours est une charge biaisée et nous ferons plus tard les mêmes démarches avec une base répondérée par les poids IPCW.

Nous vous présentons tout d'abord la méthodologie que nous avons utilisée pour trouver le seuil des graves. Nous commencerons par le graphique quantile-quantile pour identifier le domaine d'attraction de nos valeurs extrêmes. Une fois le domaine identifié, nous chercherons la valeur de $\widehat{\xi}$ (défini en section 1.3). Une fois celle-ci estimé, il nous sera possible de connaître quels outils nous pouvons utiliser pour determiner le seuil de nos valeurs extrêmes. Suite à cela, nous ajusterons une GPD puis nous nous assurerons que celle-ci colle à nos V.E.

Le QQ-plot

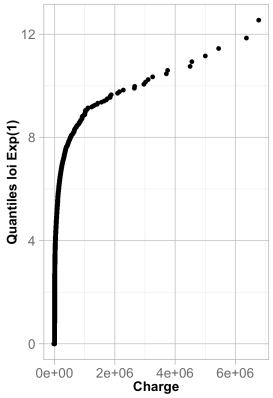
En théorie, si les observations sont issues de la loi théorique, alors les points du graphique doivent se situer le long de la première bissectrice. En pratique, à l'exception de quelques points extrêmes, la plupart des autres points doivent se situer autour de cette droite.

Dans le cas de la TVE et de la loi exponentielle, on supposera u suffisamment grand ($u = u^*$ dans notre cas) et on distinguera trois cas en fonction de la loi dont sont issues les observations et du domaine d'attraction.

Gumbel : Les points du graphique sont approximativement alignés.

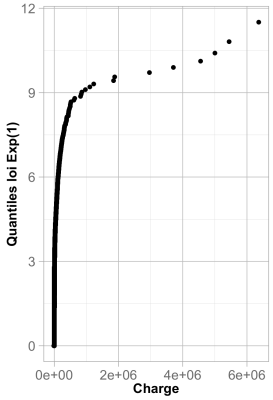
Fréchet : On observe une forme concave.

Weibull : On observe une forme convexe.

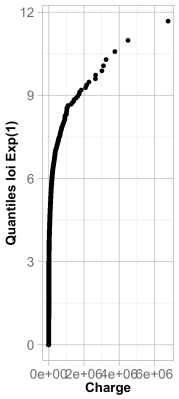


(a) 2002-2016

(b) 2002-2006



(c) 2007-2011



(d) 2012-2016

FIGURE 2.8 – QQ-plot des observations

La forme de la courbe étant concave, le QQ-plot suggère que nos extrêmes appartiennent à une loi de Fréchet. A partir de là il nous est possible d'utiliser l'estimateur de Hill afin de trouver l'estimateur de ξ.

Les estimateurs non paramétriques de l'indice de queue

Les estimateurs que nous utiliserons ici, ont été introduites dans le chapitre précédent. Nous cherchons ici à confirmer ce qu'indique le QQ-plot, c'est-à-dire que $\widehat{\xi} > 0$. De plus en fonction de la valeur l'estimateur ($\widehat{\xi} < 1$), nous pourrons en plus du Hill Plot utiliser la MEF comme méthode graphique de determination du seuil de valeurs extrêmes.

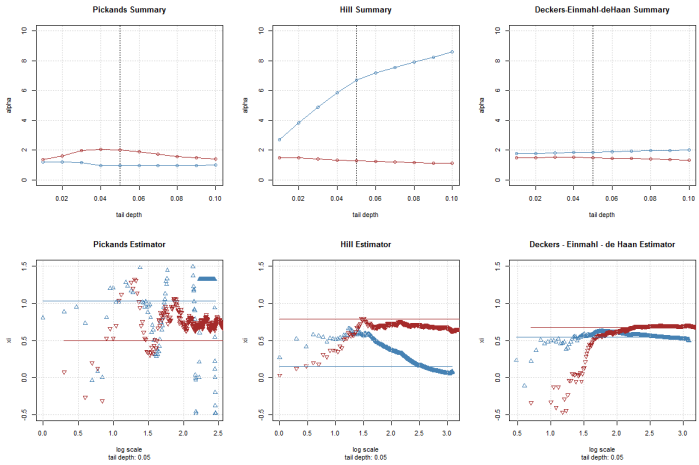


FIGURE 2.9 – Les estimateurs pour la période de 2002 à 2016

On remarque que les estimations de ξ au travers des estimateurs de Pickands, Hill et Dekkers-Einmahl sont tous compris entre 0 et 1. Nos extrêmes appartiennent donc au domaine de Fréchet, et nous pouvons utiliser deux méthodes graphiques pour déterminer le seuil, le Hill-Plot et la Mean Excess Plot.

Pour les sous-périodes, les résultats disponibles en annexe montrent une stabilité au cours du temps.

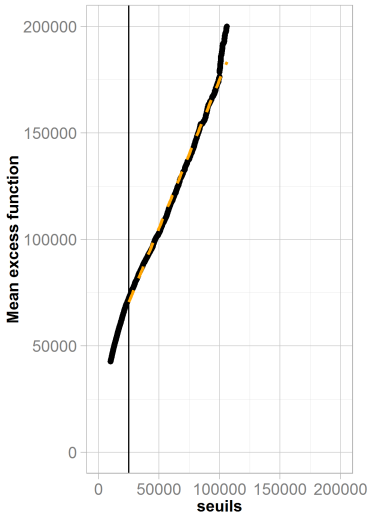
2.4.1 Estimation du seuil

La théorie des valeurs extrêmes nous fournit des outils pour estimer la loi des observations au-dessus d'un seuil bien choisi et calculer les quantiles extrêmes à l'aide de cette loi estimée. Dans la suite, on s'intéresse au comportement (statistique) des observations au-delà d'un seuil u fixé (a priori u sera choisi « grand »).

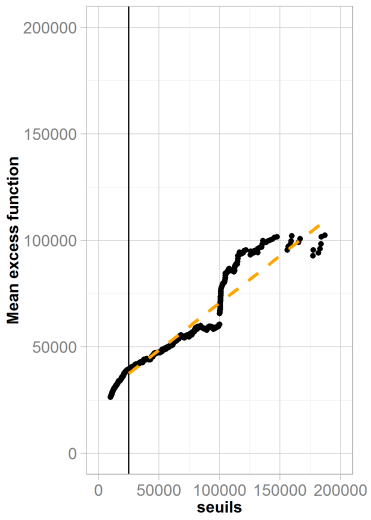
Parmi les méthodes d'estimation de seuil disponibles, nous excluons volontairement la méthode "des valeurs record", qui est légitime asymptotiquement mais qui donne des résultats aberrants dans notre cas (car nous avons un nombre fini d'observations).

Le Mean Excess Plot

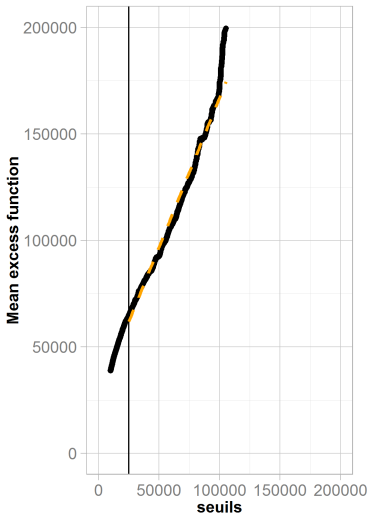
Les propriétés de la Mean Excess Function nous permettent, via les estimations de $\widehat{e}(u)$, de déterminer un seuil $u*$. De plus, comme on l'a vu, ce seuil correspond au moment où la courbe devient linéaire.



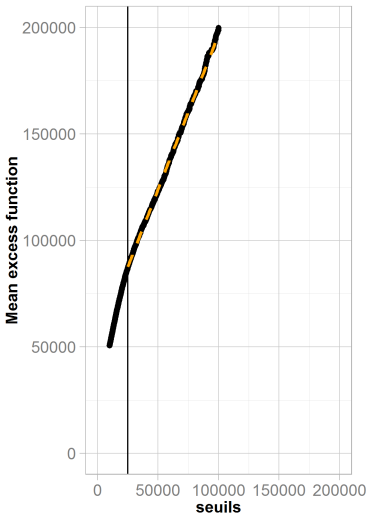
(a) 2002-2016



(b) 2002-2006



(c) 2007-2011



(d) 2011-2016

FIGURE 2.10 – MEF

A partir de ces graphiques nous remarquons trois choses :

- Sur toute la période, ainsi que sur la période 2011-2016, le seuil $u*$ est d'environ 25 000€ par victime.
- Sur la période 2007-2011, le seuil $u*$ est de 20 000€, on démontrera plus tard que cela est dû à l'inflation.
- Sur la période 2002-2006, on a une première rupture à 25 000€, puis une seconde proche des 100 000€. On devrait donc choisir le seuil de 100 000€, cependant ce paquet n'est pas vraiment significatif. En effet, on observe ici les sinistres survenus entre 2002 et 2006 et qui se

sont clôturés sur la même période. On a donc peu de graves, ce qui explique l'instabilité du graphique.

Seconde estimation du seuil par l'estimateur de Hill

Le graphe de l'estimateur de Hill (figure 2.9 et 2.11) consiste à tracer la valeur de l'estimateur en fonction de l'indice de la statistique d'ordre (k), soit l'estimateur construit sur les observations supérieures ou égales au seuil $X_{(k)}$. En haut du graphe sont indiqués les seuils $X_{(k)}$. L'estimateur de Hill est volatile lorsque k est faible, puis se stabilise : l'estimateur est déterminé comme le seuil à partir duquel l'estimateur se stabilise.

Pour la période de 2002 à 2016 Une première étape est d'estimer notre estimateur de Hill.

Par une analyse graphique, on trouve que celui-ci est proche de 0.9.

On peut non seulement utiliser le Hill plot pour retrouver cette valeur de $\widehat{\xi}$ mais aussi pour déterminer notre seuil.

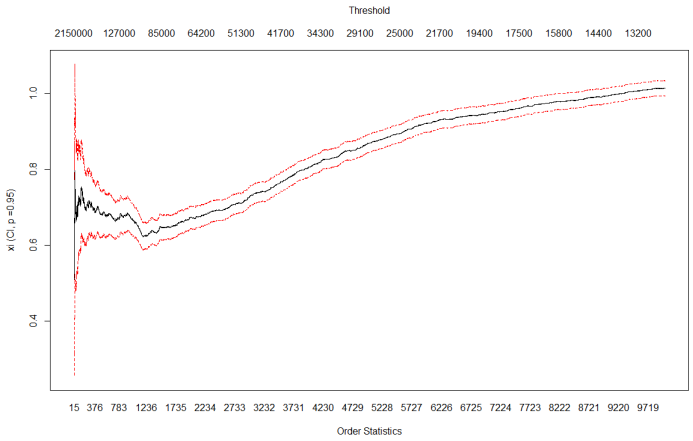


FIGURE 2.11 – Graphique de Hill

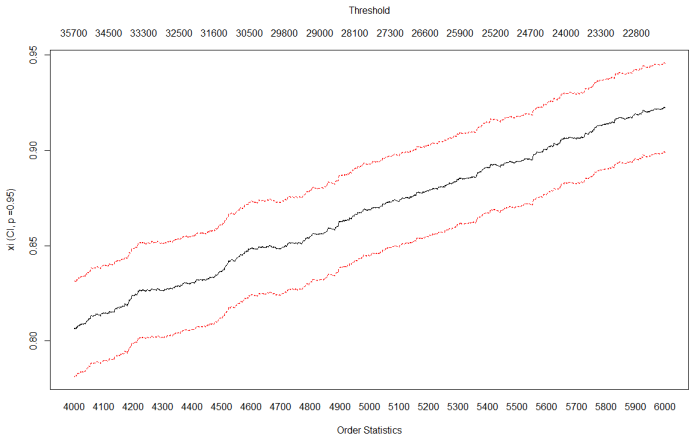


FIGURE 2.12 – Zoom du graphique de Hill

Sur le graphe obtenu, l'estimateur de Hill oscille énormément au début, puis croît et se stabilise bien : nous identifions le début de la zone de stabilité autour de 25 000. Les variations qui suivent sont d'amplitude sensiblement inférieure aux intervalles de confiance, aussi semble-t-il raisonnable de définir la zone de stabilité à partir de la fin de la tendance croissante. Un agrandissement de cette région (figure 2.12) conduit à un seuil proche de 25 000.

Une fois le seuil $\widehat{u}$ trouvé, il nous est possible de confirmer la valeur $\widehat{\xi}$ estimé en ajustant une loi de Pareto Généralisée.

2.4.2 Ajustement de la GPD

Nous avons vu précédemment la densité de la distribution GPD, que nous rappelons :

$$f_{\mu,\sigma,\xi}(x) = \frac{1}{\sigma}\left(1 + \frac{\xi(x-\mu)}{\sigma}\right)^{\left(-\frac{1}{\xi}-1\right)}$$

L'estimation des paramètres peut être effectuée par le maximum de vraisemblance.

$$log f_{\mu,\sigma,\xi}(x) = -log(\sigma) - \left(\frac{1}{\xi} - 1\right) log\left(1 + \frac{\xi(x-\mu)}{\sigma}\right)$$

μ étant le seuil de nos valeurs extrêmes, on peut poser comme précédemment $y = x - \mu$ (cf. Section 1.3.3).

$$log f_{\sigma,\xi}(y) = -log(\sigma) - \left(\frac{1}{\xi} - 1\right) log\left(1 + \frac{\xi(y)}{\sigma}\right)$$

Il ne nous reste plus qu'à estimer $\widehat{\xi}$ et $\widehat{\sigma}$.

Nous nous concentrons sur la période 2002-2016, les résultats par sous-période sont disponibles en annexe. Pour résoudre la minimisation de log-vraissemblance, nous avons utilisé la fonction gevFit du package R fExtremes [215]. Voici les résultats obtenus :

Estimateur	Paramètres estimés	Déviance	Log-Likelihood
$\widehat{\xi}$	5.411198e-01	2.297684e-02	65112.02
$\widehat{\sigma}$	3.108821e+04	1.005562e+03	

TABLE 2.3 – Statistiques de la GPD

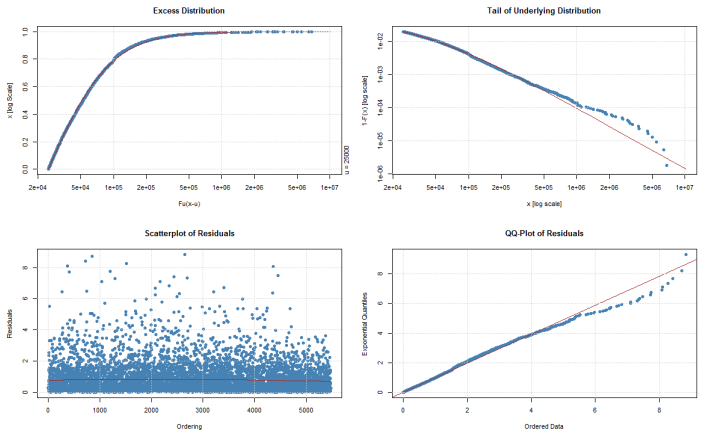


FIGURE 2.13 – GPD

On constate que la loi que l'on a ajustée fournit une adéquation raisonnable avec une GPD dont les paramètres sont dans le tableau 2.3.

2.4.3 Mise en as-if

L'inflation est un phénomène persistant qui fait monter l'ensemble des coûts et auquel se superposent des variations sectorielles de charge. En plus de l'inflation qui est liée à l'économie en elle-même (1€ de 2019 ne vaut pas 1€ en 2020) plusieurs réformes ont eu un impact sur la charge des sinistres. On pense notamment à la réforme Dintilhac [58], mais également aux différents barèmes d'indemnisation des préjudices corporels (Gazette du palais [167], Barème cour d'Appel [49] [168]) qui ont été revus plusieurs fois au cours de notre étude favorisant toujours plus l'indemnisation des victimes.

À cause de cette inflation, un sinistre clôturé en 2002 n'aura donc pas la même charge qu'un sinistre clôturé en 2016 à caractéristiques égales. On va donc corriger les montants des sinistres en fonction de leur année de clôture afin de les rendre comparables. Concernant les sinistres en cours, étant donné qu'on prend la dernière évaluation, nous n'avons pas besoin de mettre les montants à jour. Cette transformation s'appelle une mise en as-if.

L'analyse du coût moyen sur notre période nous permet de constater une inflation de $6,3\%$ par an. En poussant un peu plus l'étude, on constate que la période pré-Dinthilac a une inflation de $9,1\%$ et la période post-Dinthilac une inflation de 4%. Pour la mise en as-if, on utilisera donc la formule suivante :

$$charge\ inflate = charge * (1.091^{nb\ annee\ avant\ 2007}) * (1.039^{nb\ annee\ depuis\ 2007})$$

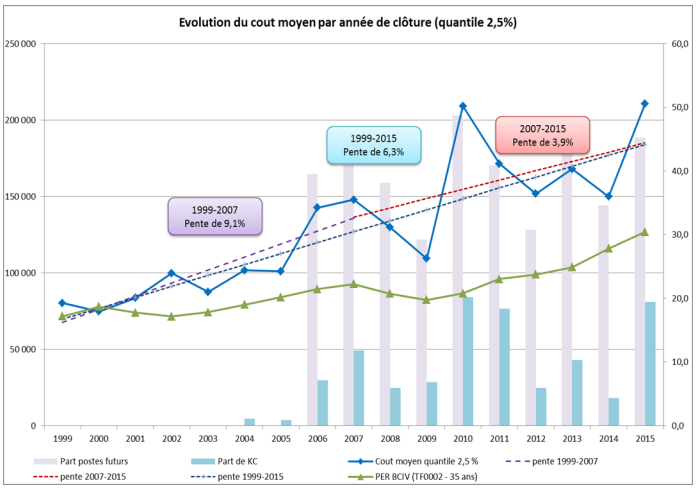


FIGURE 2.14 – Taux d'inflation appliqué à nos montants

2.4.3.1 Quantile extrême

Dans cette partie, nous allons regarder à quel quantile correspond le seuil de 25 000€ sur la période 2012-2016.

Année de clôture	92,5%	95%	97,5%
2002	788,41	1210,94	2058,96
2003	2454,36	3542,18	6714,32
2004	3873,06	5946,11	13764,62
2005	4223,22	6687,01	13764,62
2006	4932,68	8325	19376,96
2007	5029,45	7956,23	16740,71
2008	4931,57	7777,33	17520
2009	5036,72	8175,61	16958,18
2010	5671,37	9194,56	21974,47
2011	5667,85	9386,17	21774,46
2012	5402,29	8868,2	20541,1
2013	5855	9300,7	21149,54
2014	6206	10084	21969,85
2015	6676	10888	25170,36
2016	6684,36	11061,66	26760

TABLE 2.4 – Quantiles 92,5%, 95%, 97,5% par année de clôture

On remarque que le quantile 97,5% correspond au seuil des graves pour les périodes 2007-2011 (20 000€) et 2012-2016 (25 000€). En partant sur ce quantile, on va effectuer une régression linéaire afin de projeter ce quantile sur la période 2002-2006, ce qui va aussi nous permettre de lisser l'effet structure.

Année de clôture	Quantile 2,5% via la régression
2002	11 551
2003	12 552
2004	13 552
2005	14 553
2006	15 553
2007	16 554
2008	17 554
2009	18 555
2010	19 555
2011	20 556
2012	21 556
2013	22 557
2014	23 557
2015	24 557
2016	25 558

TABLE 2.5 – Quantiles 2,5% par année de clôture via régression

Et en appliquant la formule de l'inflation vue plus haut.

Année de clôture	Quantile 2,5% via la régression inflaté
2002	25 194
2003	25 093
2004	24 833
2005	24 442
2006	23 944
2007	23 358
2008	23 840
2009	24 253
2010	24 601
2011	24 889
2012	25 121
2013	25 300
2014	25 430
2015	25 515
2016	25 558

TABLE 2.6 – Quantiles 2,5% par année de clôture via régression inflaté

Le seuil de 25 000€ est à la fois cohérent et stable à travers les années.

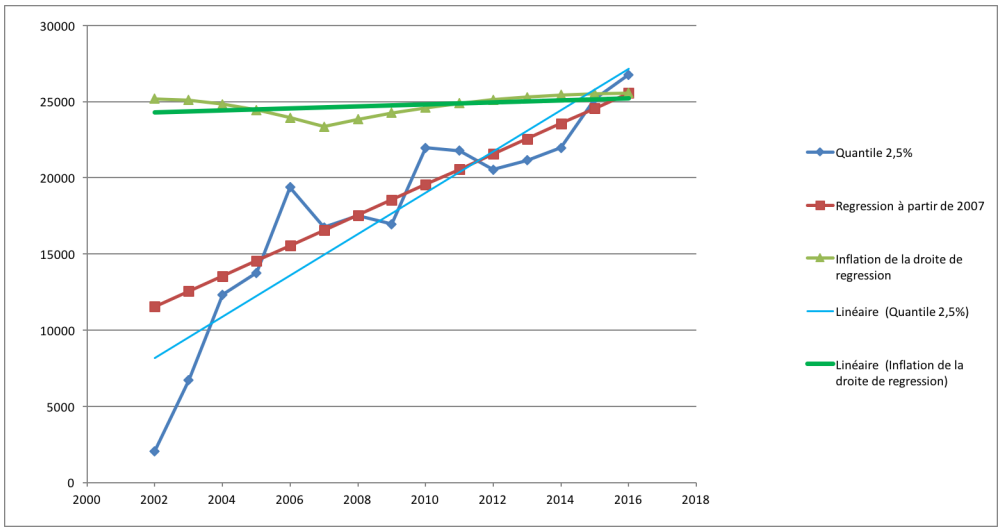


FIGURE 2.15 – Évolution du quantile 2,5% avec inflation et sans inflation

De plus, cela nous permet d'obtenir nos pentes courte période et la pente longue période définies par la formule :

$$\left(\frac{charge\ predite\ inflate}{charge\ predite}\right)^{1-nombre\ d'annee\ jusqu'\ 2016})$$

Année de clôture	coefficient pente courte période
2002	5,7%
2003	5,5%
2004	5,2%
2005	4,8%
2006	4,4%
2007	3,9%
2008	3,9%
2009	3,9%
2010	3,9%
2011	3,9%
2012	3,9%
2013	3,9%
2014	3,9%
2015	3,9%

TABLE 2.7 – Pente courte période

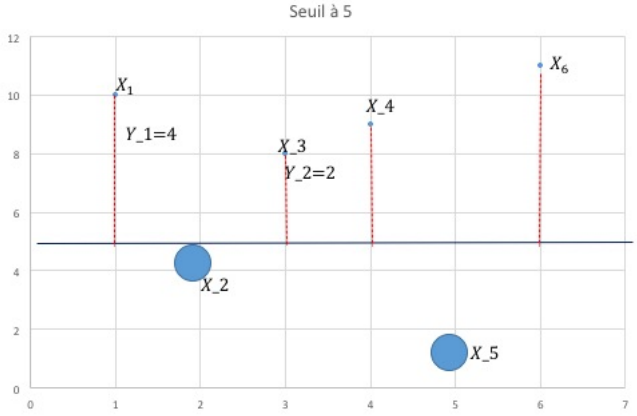
Quant à la pente longue période, on obtient un coefficient de $5,8\%$.

2.4.4 Gestion des sinistres en cours

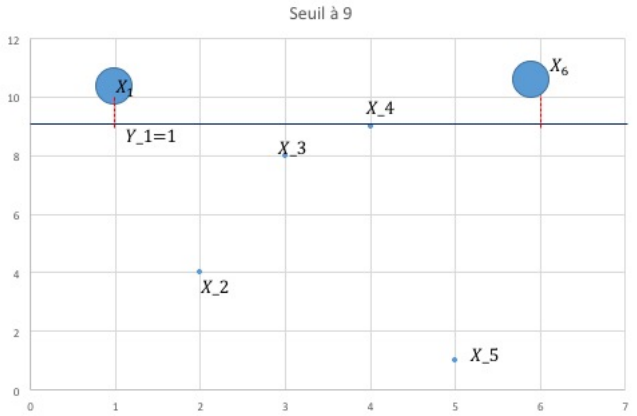
Avant d'expliquer comment nous avons décidé de gérer les sinistres en cours, nous souhaitons rappeler que nous travaillons depuis la section 2.4.3 avec des sinistres dont la charge a été inflatée en

utilisant la formule vue plus haut. Nous considérons également que les sinistres en cours ayant une évaluation constante sont vus en « euros 2016 », date de notre étude.

Comme nous avons pu le voir précédemment, la gestion des sinistres en cours est une vraie problématique. Les prendre en considération ou non reviendrait dans les deux cas à introduire un biais dans notre étude. En effet, on sous-estimerait notre seuil en excluant nos sinistres en cours, et on risque de mal estimer en cas de variation importante de nos estimations de charge.



(a) Sur-pondération des petits



(b) Sur-pondération des graves

FIGURE 2.16 – Risque d'une sur-pondération sur la méthode POT

De plus, la distribution des clos est différente de celle des en cours. On constate que la moyenne et la médiane des sinistres ouverts sont 10 fois supérieures à celles des sinistres clos.

Statistique	Clos	En cours
Min	-28982	-2501
1er Quartile	0	648
Médiane	120	1518
Moyenne	3572	36565
3e Quartile	1577	8618
Maximum	8008503	8908724
Variance	1914753097	73276903914
Std	43758	270697
Nombre	280498	25648

TABLE 2.8 – Statistiques des clos et des en cours

En plus de la moyenne et la médiane, nous avons décidé de découper la charge de nos sinistres en plusieurs intervalles :

- Inférieur à 5000€.
- De 5000€ à 50 000€ par pas de 5000€.
- De 50 000€ à 250 000€ par pas de 50 000€.
- De 250 000€ à 1000 000€ par pas de 250 000€.
- De 1000 000€ à 3000 000€ par pas de 1000 000€.
- Supérieur à 3000 000€.

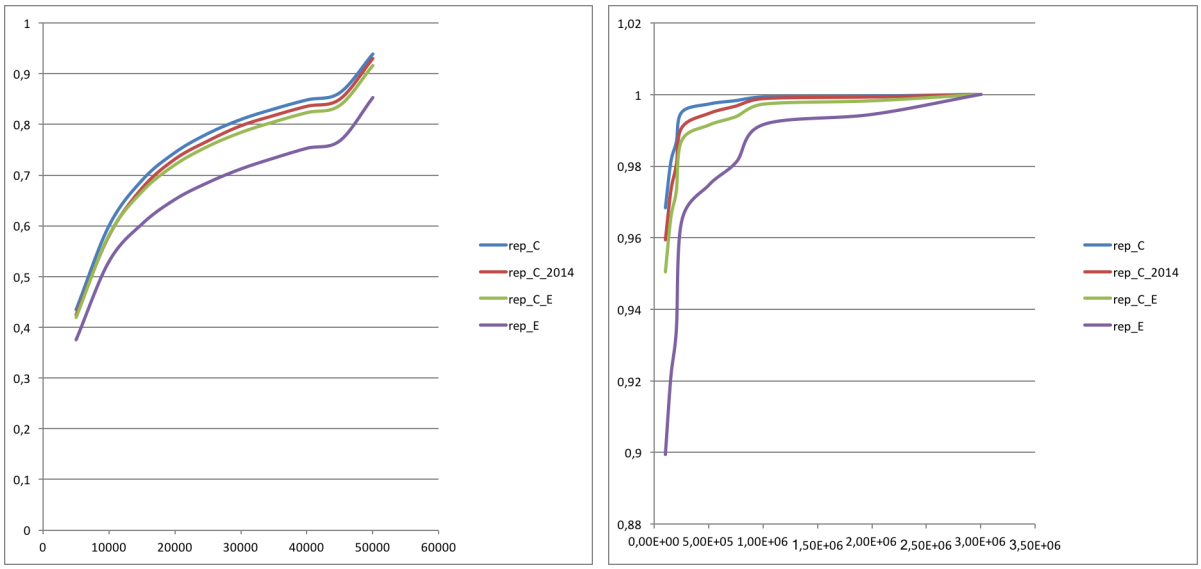
Une première idée d'étude descriptive est de regarder la répartition du ratio entre le nombre de sinistres clos et le nombre de sinistres total sur cet intervalle. On utilisera donc la formule $poids_i = \frac{nb\ sinistres_i}{nb\ clos_i}$ où i est un intervalle cité au-dessus. Nous avons constaté un ratio qui ne cessait d'augmenter au fur et à mesure qu'on allait dans la queue droite de la distribution. En effet, les sinistres qui ont une charge importante ont également une durée de vie importante. En plus de prouver une certaine corrélation entre la charge et la durée de vie du sinistre, on apprend surtout qu'on a beaucoup plus des sinistres extrêmes censurés que non censurés.

Dans un second temps, nous décidons de nous intéresser à la fonction de répartition des sinistres clos et en cours. Comme rappelé en introduction de cette thèse, il existe un lien entre la fonction de répartition et la fonction de survie qui est à l'origine des poids IPCW.

On s'intéresse volontairement aux sinistres supérieurs à 5000€. En effet, la répartition des sinistres est très spéciale en dehors de cet intervalle car près de 35% des sinistres sont sans suite.

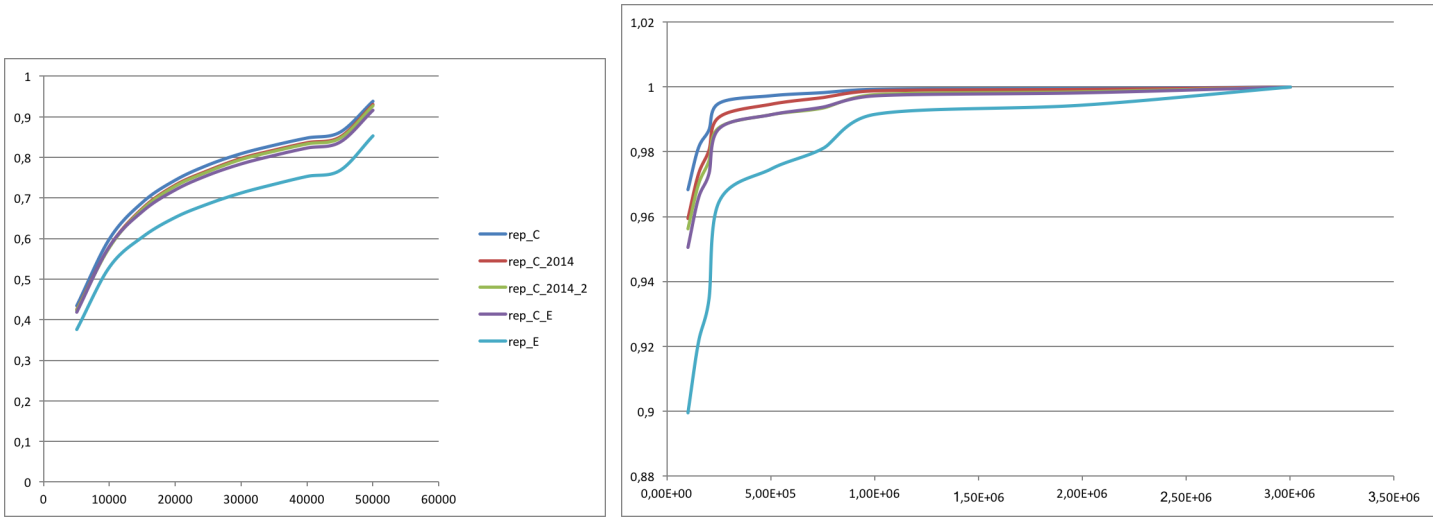
Nous décidons de prendre la fonction de répartition des sinistres clos entre 2014 et 2016 comme référence. Le choix de cette période se justifie par le fait qu'elle couvre tous les sinistres survenus entre 2002 et 2016 et qui se sont clos entre 2014 et 2016, ce qui donne un exercice représentatif.

Pour corriger le biais des en cours, nous utiliserons les poids IPCW. [213].



(a) Probabilité inférieur à 0.9 (b) Probabilité supérieur à 0.9

FIGURE 2.17 – Fonction de répartition avant pondération



(a) Probabilité inférieur à 0.9 (b) Probabilité supérieur à 0.9

FIGURE 2.18 – Fonction de répartition avant pondération

La méthodologie étant la même que précédemment, nous ne nous attarderons pas à vous la représenter de façon aussi détaillé.

Afin de trouver le seuil des valeurs extrêmes, nous estimons ξ au travers des estimateurs de Pickands, Hill et Dekkers-Einmahl.

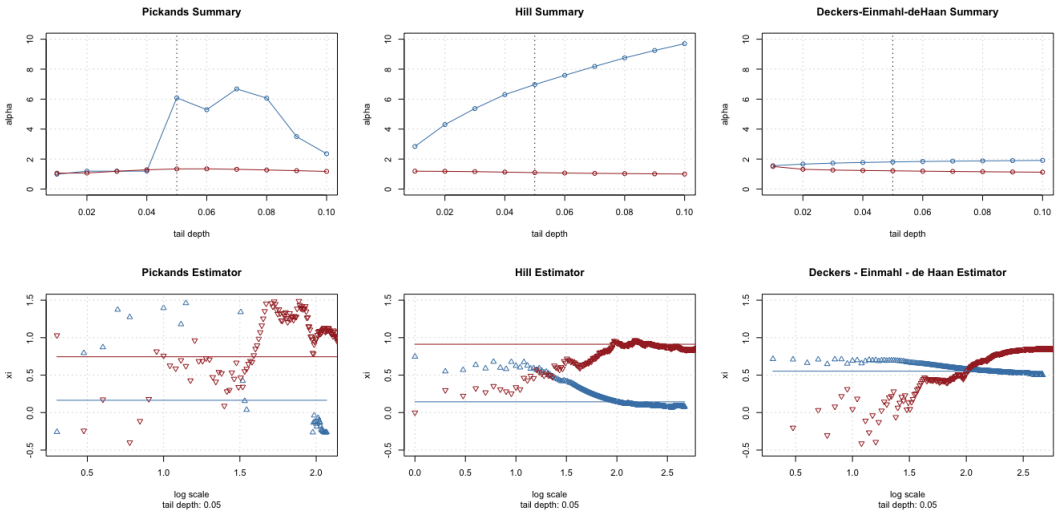


FIGURE 2.19 – Les estimateurs

Comme précédemment, on trouve $0 < \xi < 1$ et on peut donc utiliser à la fois la MEF et le Hill Plot pour trouver $\widehat{u}$.

Comme précédemment, on constate que notre seuil se trouve aux alentours des 25 000. On pose donc $u* = 25000$, et nous cherchons à estimer la loi de nos excès au delà de ce seuil.

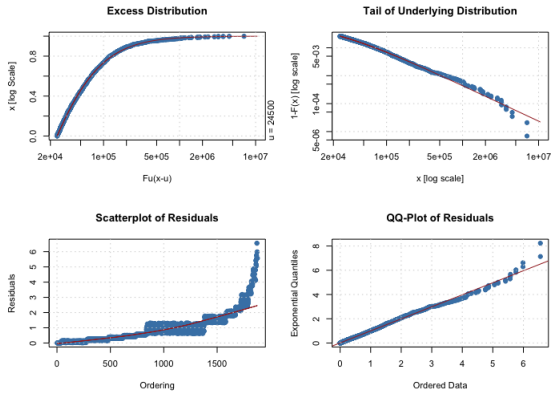
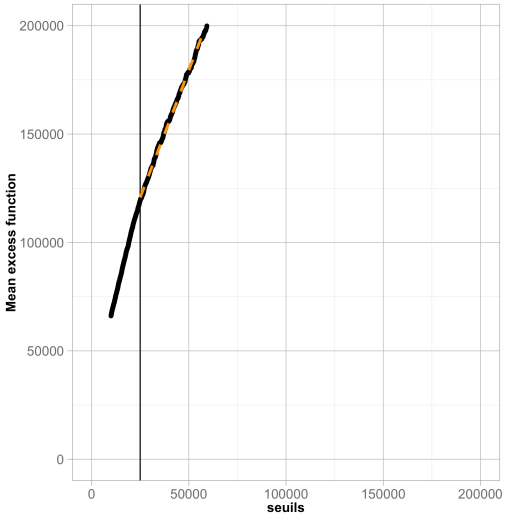


FIGURE 2.21 – GPD

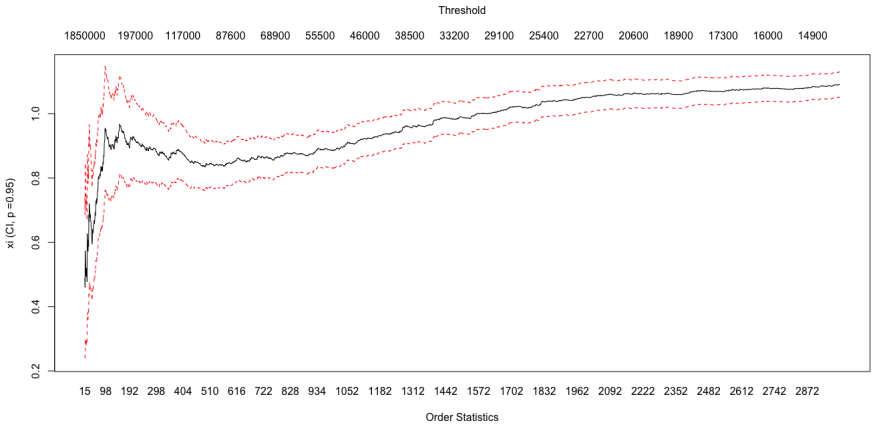
Notre GPD est en bonne adéquation avec nos extrêmes, cela valide donc le choix du seuil à 25000. De plus, les estimateurs de la GPD sont

Estimateur	Paramètres estimés	Déviance	Log-Likelihood
$\widehat{\xi}$	7.831843e-01	0.03576719	22847.14
$\widehat{\beta}$	3.250866e+04	538.49303889	

TABLE 2.9 – Statistiques de la base suite à Kaplan-Meier



(a) Mean Excess Plot



(b) Graphique de Hill

FIGURE 2.20 – Estimation du seuil des valeurs extrêmes à travers deux outils graphiques, la MEF et le Hill Plot. Les deux semblent indiquer un seuil aux alentours des 25000€.

Chapitre 3

Automatic analysis of insurance reports through deep neural networks to identify severe claims.

In this paper, we develop a methodology to automatically classify claims using the information contained in text reports (redacted at their opening). From this automatic analysis, the aim is to predict if a claim is expected to be particularly severe or not. The difficulty is the rarity of such extreme claims in the database, and hence the difficulty, for classical prediction techniques like logistic regression to accurately predict the outcome. Since data is unbalanced (too few observations are associated with a positive label), we propose different rebalance algorithm to deal with this issue. We discuss the use of different embedding methodologies used to process text data, and the role of the architectures of the networks.

3.1 Introduction

The automatization of text data analysis in insurance is a promising and challenging field, due to the importance of reports and analysis in the treatment of claims or contracts. In this paper, we focus on data coming from expert reports on a claim, but textual informations may be found also in declarations filled by policyholders themselves, or in medical questionaries used in credit insurance. Our aim is to process these reports and develop an automatic way to categorize the corresponding claims and to identify, soon after their occurrence, which one will lead to a heavy loss. Apart from the complexity of the input information, a particular difficulty is caused by the rareness of such claims (a few percent of the total database).

The rise of deep learning techniques introduces the possibility of developing an automated treatment of these data, with potential reduced costs - and, ideally, a more accurate analysis - for the insurance company, hence a potential increase of rentability. Text mining has received a lot of attention in the past decades in the machine learning literature, with many spectacular examples. Many examples can be found, for example, in [3] or [23], with applications to insurance such as [65] or [115].

As we already mentioned, our main difficulty is to focus on the prediction of relatively rare events, which is a classical specificity of the insurance business compared to other fields. Indeed, while typical claims and behavior are usually easy to understand and measure, the tail of the distribution of the losses is much more delicate, since there are quite few events in the database to evaluate it

properly. Of course, these so-called "extreme" events may, in some cases, constitute an important part of the total losses at a portfolio level.

In this paper, we consider the case of text data coming from experts in view of clustering some claims depending on their severity. Typically, the insurance company wants to separate "benign" claims from the heaviest ones in order to perform a different treatment of these two types of claims. Indeed, in terms of reserving, a different model can be used on these two types of events. Moreover, a particular attention can then be devoted to the most serious claims during their development. As described above, the difficulty is that the proportion of heavy claims is quite low compared to the other class. This imbalance is present in other potential applications, such as fraud detection. We rely on deep neural networks methodologies. Neural networks have already demonstrated to be valuable tools when it comes to evaluating claims, see for example [184] or [217]. The network architectures that we use - convolutional neural networks, see [6], and recurrent neural networks (such as Long Short Term Memory networks introduced by [99]) - are powerful tools to perform such text analysis, see for example [145] and [42]. The specificity of our approach is twofold :

- we introduce a bias correction in order to deal with the fact that claim development time may be long : indeed, our models are calibrated on a database made of claims that are closed, but also from unsettled ones. Calibrating the model on the closed claims solely would introduce some bias due to the over-representation of claims with small development time (this is a classical problem linked to the concept of censoring in survival analysis, see e.g. [136],[78]).
- once this bias correction has been done, a re-equilibration of the database is required to compensate the rareness of the events we focus on. For this, we use Bagging methodologies (for Bootstrap Aggregating, see [31]), introducing constraints in the resampling methodology which is at the core of Bagging.

The contribution of this paper is to show how performing these two tasks (censoring correction and re-equilibration) can improve existing deep learning methods to analyse a corpus of texts which is very specific : indeed, apart from the rareness of the event that are considered, the reports on car insurance claims that we study in our applications, are characterized by a very specific vocabulary and a particular syntax that makes text classification more challenging.

The rest of the paper is organized as follows. In section 3.2 we describe the general framework and the general methodology to address these two points. A particular focus on processing text data is shown in section 4.4, where we explain the process of embedding such data (this means to perform an adapted dimension reduction to represent efficiently such complex unstructured data). The deep neural networks architectures that we use are then described in section 3.4. In section 3.5, a real example on a database coming from a French insurer gives indications on the performance of the methodology.

3.2 Framework and neural network predictors

In this section, we describe the generic structure of our data. Our data are subject to random censoring, as a time variable (time of settlements of the claims) is present in the database. The correction of this phenomenon, via classical methodologies from survival analysis, is described in section 3.2.1. Next the general framework of neural networks is defined in section 3.2.2. The imbalance in the data, caused by the fact that we focus on predicting rare events, is described in section 3.2.3, along with methodologies to improve the performance of our machine learning approaches under this constraint.

3.2.1 The censoring framework

Our aim is to predict a 0-1 random response I, which depends on a time variable T and covariates $\mathbf{X} \in \mathscr{X} \subset \mathbb{R}^d$. In our practical application, I is an indicator of the severity of a claim ($I = 1$ if the claim is considered as "extreme", 0 otherwise). The definition of what we refer to as an extreme claim depends on management conventions that will be described in the real data section. The variable T represents the total length of the claim (i.e. the time between the claim occurence and its final settlement), and $\mathbf{X}$ some characteristics.

In our setting, the variables I and T are not observed directly. Since T is a duration variable, it is subject to right-censoring. This means that, for the claims that are still not settled, T is unknown. On the other hand, I is known only if the claim is closed. This leads to the following observations $(J_i, Y_i, \delta_i, \mathbf{X}_i)_{1\leq i\leq n}$ assumed to be i.i.d. replications of $(J, Y, \delta, \mathbf{X})$ where

$$\left\{ \begin{array}{rcl} Y & = & \inf(T, C), \\ \delta & = & \mathbf{1}_{T\leq C}, \\ J & = & \delta I, \end{array} \right.$$

and $\mathbf{X}$ represents some covariates that are fully observed (in our case, $\mathbf{X}$ is text data coming from reports, hence it can be understood as a vector in $\mathbb{R}^d$ with d being huge : more precisely, $\mathbf{X}$ can be understood as a matrix, each column representing an appropriate coding of a word, see section 4.4, the number of columns being the total number of words). The variable C is called the censoring variable and is assumed to be random. The censoring variable corresponds to the time after which the claim stops to be observed for any other reason than its settlement (either because the observation period ends before settlement, or because there has been a retrocession of the claim). This setting is similar to the one developed by [138] or [136] in the case of claim reserving, with a variable I that may not be binary in the general case. In the following, we assume that T and C have continuous distributions so that these variables are perfectly symmetric, which means that one has the same level of information on the distribution of T as on the distribution of T (otherwise, a dissymmetry can appear due to ties, since one could have $\mathbb{P}(T = C) \neq 0$, leading to potential inconsistencies, see [196]).

Considering only the complete data (i.e. with $\delta_i = 1$, which corresponds to the case where the claim is settled) would introduce some bias : among the complete of observations, there is an overrepresentation of claims associated with a small value of T_i. Since T and I are dependent (in practice, one often observes a positive correlation between T and the corresponding amount of the claim), neglecting censoring would imbalance the sample, modifying the proportion of values of i for which $I_i = 1$.

Our aim is to estimate the function $\pi(\mathbf{x}) = \mathbb{P}(I = 1|\mathbf{X} = \mathbf{x})$, in a context where the dimension d of $\mathbf{x}$ is large. Typically, how an estimated function fits data is quantified by minimizing some loss function. If we were dealing with complete data, a natural choice would be the logistic loss function (also called "cross entropy" in the neural network literature, see for example [114]), that is

$$L_n^*(p) = -\frac{1}{n}\sum_{i=1}^{n} \left\{ I_i \log p + (1 - I_i) \log(1 - p) \right\},$$

which is a consistent estimator of

$$L(p) = -E\left[I \log p + (1 - I) \log(1 - p) \right].$$

As stated before, computation of L_n^* is impossible in presence of censoring due to the lack of information, and an adaptation is required to determine a consistent estimator of L.

A simple way to correct the bias caused by censoring is to rely on the Inverse Probability of Censoring Weighting (IPCW) strategy, see for example [177]. The idea of this approach is to find a function $y \to W(y)$ such that, for all positive function ψ with finite expectation,

$$E\left[\delta W(Y)\psi(J,Y,\mathbf{X})\right] = E\left[\psi(I,T,\mathbf{X})\right], \tag{3.1}$$

and to try to estimate this function W (in full generality, this function can also depend on $\mathbf{X}$). The function W depends on the identifiability assumptions that are required to be able to retrieve the distribution of I and T from the data. In the following, we assume that

1 C is independent from (J,T)

2 $\mathbb{P}(T \leq C|I,T,\mathbf{X}) = \mathbb{P}(T \leq C|T)$,

as in [136], and is similar to the one used, for example, in [198] or [199] in a censored regression framework. These two conditions ensure that (3.1) holds to $W(y) = S_C(y)^{-1} = \mathbb{P}(C \geq y)^{-1}$, provided that $\inf\{t : S_C(t) = 0\} \geq \inf\{t : \mathbb{P}(T \geq t) = 0\}$, which will be assumed throughout this paper. These assumptions are identifiability assumptions that can not be tested statistically, see [7], section III.2.2.3 p.148. Alternative assumptions can be found for example in [208], and the procedure we develop can be modified to address this more complex situation, in the same spirit as [78]. In the real data application that we consider in Section 3.5 below, censoring is mainly caused by administrative issues (the claim is not closed at the date of extraction of the database), which makes these more classical identifiability assumptions reasonable. This also ensures that W is strictly positive and finite.

S_C can then be consistently estimated by the Kaplan-Meier estimator (see [112]), that is

$$\widehat{S}_C(t) = \prod_{Y_i \leq t} \left(1 - \frac{\delta_i}{\sum_{j=1}^n \mathbb{1}_{Y_j \geq Y_i}}.\right)$$

Consistency of Kaplan-Meier estimator has been derived by [197] and [5].

Hence, each quantity of the type $E\left[\psi(I,T,\mathbf{X})\right]$ can be estimated by $\sum_{i=1}^n W_{i,n}\psi(J_i,Y_i,\mathbf{X}_i)$, where

$$W_{i,n} = \delta_i n^{-1} \widehat{S}_C(Y_i)^{-1}. \tag{3.2}$$

In other words, a way to correct the bias caused by the censoring consists in attributing the weight $W_{i,n}$ at observation i. This weight is 0 for censored observations, and the missing mass is attributed to complete observations with an appropriate rule. A convenient way to apply this weight once and for all is to duplicate the complete observations in the original data accordingly to their weight. This is a way to artificially (but consistently) compensate the lack of complete observations associated with a large value of Y_i (let us observe that $W_{i,n}$ is an increasing function of Y_i). This leads to the following duplication algorithm.

Algorithm 3 Duplication algorithm for censored observations

Require:

- a dataset $(J_i, \mathbf{Y}_i, \delta_i, \mathbf{X}_i)_{1\leq i\leq n}$.
- a list of weights $W_{i,n}$ computed from the data accordingly to (D.2), and let $w = \min_{1\leq i\leq n} W_{i,n}$.

Set $n' = 1$.
for $i \leftarrow 1$ **to** n **do**
1. If $\delta_i = 0$, go to step $i + 1$.
2. Else :
   - Let $k = \lfloor W_{i,n}/w \rfloor$, where $\lfloor x \rfloor$ denotes the integer part of x.
   - Define, for $j \in \{1, ..., k\}$ $J'_{n'-1+j} = J_i$, $Y'_{n'-1+j} = Y_i$, $\mathbf{X}_{n'-1+i} = \mathbf{X}_i$.
   - $n' \leftarrow n' + k$.

end for

return The duplicated dataset : $(\mathbf{Z}_i)_{1\leq i\leq n'} = (J'_i, Y'_i, \mathbf{X}'_i)_{1\leq i\leq n'}$.

The cross-entropy that we want to minimize is then computed from the duplicated dataset, defining

$$L_{n'}(p) = -\frac{1}{n'}\sum_{i=1}^{n'} \left\{ J'_i \log p + (1 - J'_i) \log(1 - p) \right\}. \tag{3.3}$$

Remark. In the experimental part of the paper, we will also introduce two alternative loss functions, which can be seen as modifications of cross-entropy. The focal-loss (see [133]) is defined as

$$\tilde{L}_{n'}(p) = -\frac{1}{n'}\sum_{i=1}^{n'} \left\{ J'^{\gamma}_i \log p + (1 - J'_i)^{\gamma} \log(1 - p) \right\}. \tag{3.4}$$

where $\gamma \geq 0$. The parameter γ helps to focus on "hard" examples, by avoiding attributing too much weights at easily well-predicted observations. The Weighted Cross Entropy (see [156]) is a weighted version of (4.3). Introducing a weight $\alpha \in (0, 1)$, define

$$\bar{L}_{n'}(p) = -\frac{1}{n'}\sum_{i=1}^{n'} \left\{ J'_i \alpha \log p + (1 - \alpha)(1 - J'_i) \log(1 - p) \right\}. \tag{3.5}$$

The parameter α can be used to increase the importance of good prediction for one of the two classes, or to reflect its under-representation in the sample.

3.2.2 Neural network under censoring

A neural network (see for example [96], chapter 11) is an over-parametrized function $\mathbf{x} \to p(\mathbf{x}, \theta)$, that we will use in the following to estimate function π. By over-parametrized, we mean that $\theta \in \mathbb{R}^k$, with k usually much larger than d (the dimension of $\mathbf{x}$). The function $p(\cdot, \theta)$ can also have a complex structure depending on the architecture of the network, that we present in more details in section 3.4. Typically, a neural network is a function that can be represented as in Figure 4.1. Following the notations of Figure 4.1, each neuron is described by a vector of weights $\mathbf{w}$ (of the same size as $\mathbf{x}$), a bias term b (1-dimensional), and an activation function f used to introduce some nonlinearity.

Typical choices of functions f are described in Table C.1. Following our notations, θ represents the list of the weights and bias parameters corresponding to all neurons of the network. For a given value of θ and an input $\mathbf{x}$, the computation of $p(\mathbf{x}, \theta)$ is usually called "forward propagation".

On the other hand, "backward propagation" (see [181]) is the algorithmic procedure used to optimize the value of the parameter θ. Since our aim is to estimate the function π, fitting the neural network consists of trying to determine

$$\theta^* = \arg\min_{\theta} L_{n'}(p(\cdot, \theta)),$$

where we recall that L_n has been defined in equation (4.3) (or, alternatively, using $\tilde{L}_{n'}$ or $\bar{L}_{n'}$ from (3.4) and (4.4)), and is adapted to the presence of censoring since computed on the duplicated dataset obtained from Algorithm 7.

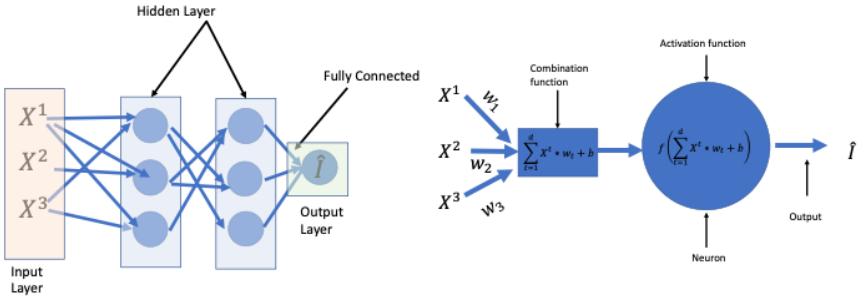


FIGURE 3.1 – Left-hand side : representation of a simple neural network (multilayer perceptron). Each unit of the network (represented by a circle), is called a neuron (a neuron is represented on the right-hand side). $X^{(j)}$ represents the $j-$th component of the covariate vector $\mathbf{X}$.

3.2.3 Bagging for imbalanced data

Bagging (for bootstrap aggregating, see [31]) is a classical way to stabilize machine learning algorithms. The heuristic behind bagging is to avoid overfitting, and to perform a synthesis (through the aggregation of results) of different calibrations of the models, which stabilizes the results. In an ideal situation, one would use aggregation of models calibrated on totally different datasets generated from the same distribution as the one we are targeting through our statistical analysis. This is of course impossible, since generating the observed random sample is an experiment that can not be replicated indefinitely, so pseudo-samples are generated by bootstrap, using the empirical distribution of the data instead of the true (unknown) one. Instead, bootstrap resampling is used to create pseudo-samples. In our case, a different neural network is fitted from each pseudo-sample, before aggregating all the results. Introducing bootstrap is expected to reduce the variance of estimation, while avoiding overfitting.

Formally, the procedure can be described in the following way.

Algorithm 4 Bagging algorithm applied to neural networks

Require:

- a neural network architecture, that is a function $\theta \to p(\cdot, \theta)$;
- a subset of the original data $(Z'_i)_{1 \leq i \leq m} = (Z_{\sigma(i)})_{1 \leq i \leq m}$ for $m \leq n$ and σ a permutation of $\{1, ..., n\}$;
- B the number of iterations of the algorithm (number of estimators that are aggregated).

for $k \leftarrow 1$ **to** B **do**

1. Draw a bootstrap sample $(Z_i^{(k)})$ from $(Z'_i)_{1 \leq i \leq m}$.
2. Fit $p(\cdot, \widehat{\theta}(k))$ on this bootstrap sample.

end for

return The final estimator is $\mathbf{x} \to \widehat{\pi}(\mathbf{x}) = \frac{1}{B} \sum_{k=1}^{B} p(\mathbf{x}, \widehat{\theta}^{(k)})$.

A particular difficulty in applying bagging to our classification problem is that we have a problem in which a class is under-represented among our observations. If we were to apply bagging as in Algorithm 4 directly, many bootstrap samples may contain too few observations from the smallest class to produce an accurate prediction. This could limit considerably the performance of the resulting estimator.

For structured data, methods like SMOTE (see [39]) can be used to generate new observations to enrich the database. Data augmentation techniques are also available for image analysis, see [175]. These techniques are difficult to adapt to our text analysis framework, since they implicitly rely on the assumption that there is some spatial structure underlying data : typically, if the observations are represented as points in a multi-dimensional space, SMOTE joins two points associated with label 1, and introduce a new pseudo-observation somewhere on a segment joining them. In our situation, labels are scattered through this high-dimensional space, and the procedure does not seem adapted.

We here propose two simple ways to perform rebalancing, that we refer to as "Balanced bagging" and "Randomly balanced bagging" respectively.

Balanced bagging :

Algorithm 5 Balanced Bagging Algorithm

Require:

- same requirements as in Algorithm 4.
- $n_1 =$ number of desired observations in the under-represented class (corresponding to $J_i' = 1$, denoted $\mathscr{C}_1$).

for $k \leftarrow 1$ **to** B **do**

1. Draw a bootstrap sample $(Z_i^{(k)})_{1 \leq i \leq 2n_1}$ from $(Z_i')_{1 \leq i \leq m}$, under the constraint that $(Z_i^{(k)})_{1 \leq i \leq n_1} \in \mathscr{C}_1$ and $(Z_i^{(k)})_{n_1+1 \leq i \leq 2n_1} \in \mathscr{C}_0$ (where $\mathscr{C}_0$ denotes the class of observations such that $J_i' = 0$).
2. Fit $p(\cdot, \widehat{\theta}^{(k)})$ on this bootstrap sample.
3. Compute the corresponding fitting error e_k, and let $w_k = e_k^{-1}$.

end for

return The final estimator is $\mathbf{x} \to \widehat{\pi}(\mathbf{x}) = \frac{1}{\sum_{j=1}^{B} w_j} \sum_{k=1}^{B} w_k p(\mathbf{x}, \widehat{\theta}^{(k)})$.

Randomly balanced bagging :

Algorithm 6 Randomly Balanced Bagging Algorithm

Require:

- same requirements as in Algorithm 4.
- $n_1 =$ mean number of desired observations in the under-represented class.
- a such that $n_1 + a$ is less than the total number of observations in $\mathscr{C}_1$.

for $k \leftarrow 1$ **to** B **do**

1. Simulate two independent variables (U_0, U_1) uniformly distributed on $[-a, a]$, and define $\widehat{n}_0 = n_1 + U_0$, $\widehat{n}_1 = n_1 + U_1$.
2. Draw a bootstrap sample $(Z_i^{(k)})_{1 \leq i \leq \widehat{n}_0 + \widehat{n}_1}$ from $(Z_i')_{1 \leq i \leq m}$, under the constraint that $(Z_i^{(k)})_{1 \leq i \leq \widehat{n}_1} \in \mathscr{C}_1$ and $(Z_i^{(k)})_{\widehat{n}_1 + 1 \leq i \leq \widehat{n}_1 + \widehat{n}_0} \in \mathscr{C}_0$.
3. Fit $p(\cdot, \widehat{\theta}^{(k)})$ on this bootstrap sample.
4. Compute the corresponding fitting error e_k, and let $w_k = e_k^{-1}$.

end for

return The final estimator is $\mathbf{x} \to \widehat{\pi}(\mathbf{x}) = \frac{1}{\sum_{j=1}^{B} w_j} \sum_{k=1}^{B} w_k p(\mathbf{x}, \widehat{\theta}^{(k)})$.

In each case, the final aggregated estimator perform the synthesis of each step, giving more importance to step k if its fitting error e_k is small.

3.2.4 Summary of our methodology

Let us summarize our methodology, which combines bias correction (censoring correction), and bagging strategies to compute our predictor. First, we compute Kaplan-Meier weights and use them to duplicate data according to these weights as described in Algorithm 7. Then, from a given neural network architecture, we use one of the bagging algorithm of section 3.2.3 to create pseudo-samples, and aggregate the fitted networks to obtain our final estimator $\widehat{\pi}(\mathbf{x})$.

The quality of the prediction strongly depends on two aspects that have still not been mentioned in this general description of the method :

- **Pre-processing of text data :** this step consists in finding the proper representation of the words contained in the reports in a continuous space, adapted to defining a proper distance between words.
- **Architecture :** the performance is strongly dependent on the type of neural networks used (number of layers, structure of the cells and connexions).

This two points are developed in section 4.4 and 3.4 respectively.

3.3 Embedding methodology

In this section, we discuss the particularity of dealing with text data. To be analyzed by neural networks structures, text data first require to be transformed into covariates of large dimension in an appropriate space. After discussing one-hot encoding in section 3.3.1, we will explain the concept

of word embedding in section 4.4.2. A combination of the standard Fasttext embedding methodology (see [109]) with our database of text reports is discussed in section 3.3.2.1.

3.3.1 One-hot encoding

Since our procedure aims at automatically treating text data and use them to predict the severity of a claim, we need to find a convenient representation of texts so that it can be treated efficiently by our network. A first idea is to rely on one-hot encoding. The set of whole words in the reports forms a dictionary of size V (Vocabulary size), then each word is associated to a number $1 \leq j \leq V$. A report is then represented as a matrix. If there is l words in the report, we encode it as a matrix $\mathbf{X} = (x_{a,b})_{1 \leq a \leq V, 1 \leq b \leq l}$, with V lines and l columns. The $k-$th column represents the $k-$th word of the report, say $\mathbf{x}_k$. If this word is the $j-$th word of the dictionary, then $x_{j,k} = 1$ and $x_{i,k} = 0$ for all $i \neq j$. Hence, each column has exactly one non-zero value.

To introduce some context of use of the words, the size of the dictionary can be increased by not considering single words but $n-$grams. A n-gram is a contiguous sequence of n items from a given sample of text or speech (either a sequence of words, or of letters). The vocabulary size V becomes the total number of $n-$grams contained in the report.

Using one-hot encoding makes the data ready for machine learning treatment, but it raises at least two difficulties :

- the vocabulary size V is huge.
- it is not adapted to defining a convenient distance between words. Indeed, we want to take into account that there are similarities between words (synonyms, for instance, or even spelling errors).

Hence, there is a need for a more compact representation that would define a proper metric on words.

3.3.2 Word embedding

This process of compacting the information is called word embedding, see [26], [110]. The idea of embedding is to map each column of a one-hot encoded matrix to a lower dimension vector (whose components are real numbers, not only 0-1 valued).

Embedding consists in defining dense vector representations of the characters. The objectives of these projections are :

- Dimension Reduction : as already mentioned, embedding maps a sparse vector of size V to a lower dense vector of size $N < V$.
- Semantic Similarity : typically, words that are similar or that are often associated, will be close in terms of embedded vectors.

To perform embedding, we will use a linear transform of vectors. This transformation is defined by the embedding matrix W (V columns and N lines). A one-hot encoded report $\mathbf{X}$ is then transformed into $\mathbf{X}' = W\mathbf{X}$ (according to the notations of section 3.2), which becomes the input of our neural network models. How to determine a proper matrix W is described by the diagram of Figure 3.2. A full description of the steps of the methodology is the following :

— a "context window" of size $c = 2p + 1$ is centered on the current word $\mathbf{x}_k$: this defines a set of c words $(\mathbf{x}_{k-j})_{-p \leq j \leq p}$ that are used as inputs of the procedure of Figure 3.2.

- the embedding matrix W is applied to each of these word, and an hidden layer, producing, for the $k-$th group of words, $h_k = C^{-1} \sum_{j=-p}^{p} W\mathbf{x}_k$.
- the $N-$dimensional vector h_k is then sent back to the original $V-$dimensional space by computing $\widehat{\mathbf{x}}_k = W' h_k = (\widehat{x}_{j,k})_{1 \leq j \leq V}$, where W' denotes the synaptic matrix of the hidden layer also called the "context embedding".

The error used to measure the quality of the embedding for word k is

$$-\sum_{j=1}^{V} \log \left(\frac{\exp(x_{j,k}\widehat{x}_{j,k})}{\sum_{l=1}^{V} \exp(x_{l,k}\widehat{x}_{l,k})} \right).$$

A global error is then obtained by summing all the words of the report, and by summing over all the reports. The embedding matrix is the one which minimizes this criterion. In this paper, we consider embedding through neural networks, but alternative methodologies such as [161] can be used.

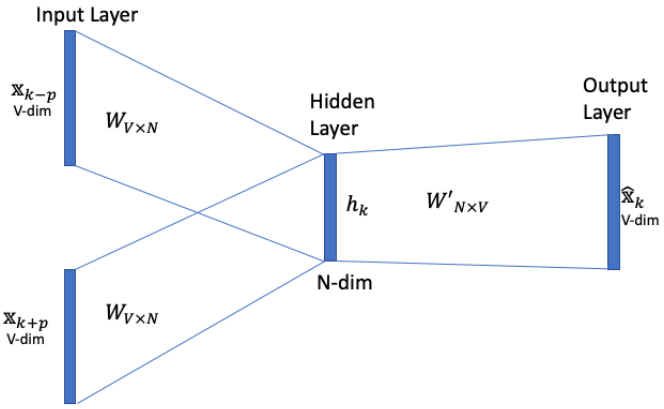


FIGURE 3.2 – Summary of the embedding matrix determination.

3.3.2.1 FastText

The gist of FastText, see [109], is that instead of directly learning a vector representation for a word (as with Word2vec, see [147], [174]), the words are decomposed into small character n-grams (see [27]), and one learns a representation for each n-gram. A word is represented as words being represented as the sum of these representations. So even if a word is misspelled, its representation tends to be close to the one of the well-spelled word. Moreover, FastText proposes an efficient algorithm to optimize the choice of the embedding matrix. A pre-trained FastText embedding matrix is available, trained on a huge number of words. Nevertheless, it has not been calibrated from on a vocabulary and a context specific to insurance. Hence, a possibility that we develop in the following is to use the FastText algorithm (not the pre-trained matrix), but training our embedding matrix on the corpus of insurance reports from our database.

In the following, we distinguish between three ways of performing embedding :

- the matrix W is determined via FastText's algorithm once and for all at the beginning of the procedure. The embedded words are sent as inputs into the network architectures described in the following section 3.4. Only the parameters of these networks are iteratively optimized. We call this method "static" embedding.

— the embedding is considered as a first (linear) layer of the network. The values of the embedding matrix W are initialized with FastText's algorithm, but updated at the same time the other coefficients of the network are optimized. We call this method "non-static".
— for comparison, we also consider the case of a random initialization of this matrix W, whose coefficients are then updated in the same way as in the non-static method. We refer to it as the "random" method.

3.4 Network architectures for text data

The architecture of neural networks is a key element that can considerably impact their performance. We consider to types of architectures in the following :
— Convolutional Neural Networks (CNN), described in section 3.4.1 ;
— Long Short Term Memory (LSTM) networks which are a particular kind of Recurrent Neural Networks (RNN), described in section 3.4.2.

3.4.1 Convolutional Neural Network

Convolutional Neural Networks are nowadays widely used for image processing (image clustering, segmentation, object detection...), see for example [121]. [113] showed that they can achieve nice performance on text analysis. Compared to the multilayer perceptron, which is the most simple class of neural network and corresponds to the architecture described in Figure 4.1, CNN are based on a particular spatial structure that prevents overfitting.

A CNN is made of different type of layers composing the network. One may distinguish between :
— convolutional layers : each neuron only focus on a localized part of the input, see below ;
— fully connected layers : every neuron of the layer is connected to every neuron in the following one ;
— normalization layers : used to normalize the inputs coming from the previous layer ;
— padding and pooling layers : the aim is either to expand or to reduce the dimension of the inputs coming from the previous layer through simple operations, see below ;
— the final output layer that produces a prediction of the label through combination of the results of the previous layer.

The convolutional layers are at the core of the idea of spatial structure of CNNs. They extract features from the input while preserving the spatial relationship between the coordinates of the embedding matrix. Typically, they operate simple combinations of coefficients whose coordinates are closed to each other, as illustrated in Figure 3.3.

In CNN terminology, the matrix B in Figure 3.3 is called a 'filter' (sometimes 'kernel' or 'feature detector') and the matrix formed by sliding the filter over the input and computing the dot product is called the 'Convolved Feature' ('Activation Map' or 'Feature Map'). A filter slides over the input (denoted by A in Figure 3.3) to produce a feature map. Each coefficient of the output matrix (C in Figure 3.3) is made of a linear combination of the coefficients of A is a small square of the input matrix. This linear combination is a convolution operation. Each convolution layer aims to capture local dependencies in the original input. Moreover, different filters make appear different features, that is different structures in the input data. In practice, a CNN learns the values of these filters on its own during the training process (although we still need to specify parameters such as number of filters, filter size, architecture of the network etc. before the training process). The higher the

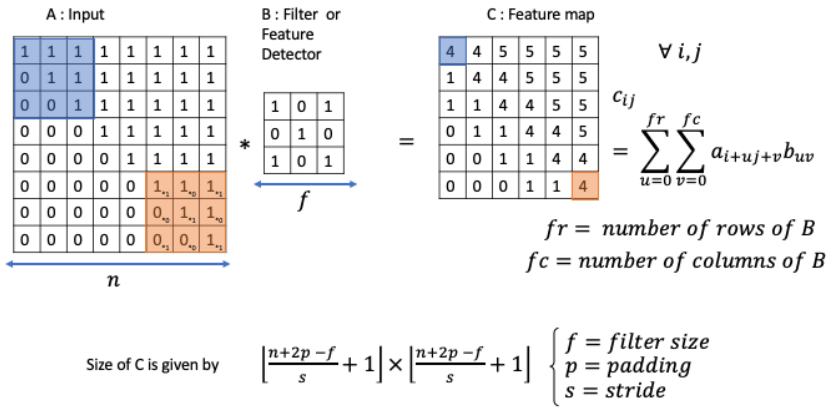


FIGURE 3.3 – An example of a convolution layer. Convolution Layer. Each coefficient $c_{i,j}$ of the output matrix is obtained by a linear combinations of coefficients $a_{i,j}$ in a $f \times f$ square of the input matrix. The color code shows which coefficients of the input matrix have been used to compute the corresponding coefficient of C.

number of filters, the more features get extracted, increasing the ability of the network to recognize patterns in unseen inputs.

The other types of layers are described in section C.0.2. An example of CNN architecture is described in the real data application, see Figure 3.7.

3.4.2 Recurrent Neural Networks and Long Short Term Memory (LSTM) networks

Recurrent Neural Networks (RNN) are designed to handle dependent inputs. This is the case when dealing with text, since a change in the order of words induce a change of meaning of a sentence. Therefore, where conventional neural networks consider that input and output data are independent, RNNs consider an information sequence. A representation of a RNN is provided in Figure 3.4.

According to this diagram, the different (embedded) words (or $n-$grams) constituting a report, pass successively into the network A. The $t-$h network produces an output (the final one constituting the prediction), but also a memory cell c_t which is sent to the next block. Hence, each single network uses a regular input (that is an embedded word), plus the information coming from the memory cell. This memory cell is the way to keep track on previous information. The most simple way to keep this information would be to identify the memory cell with the output of each network. Hence, the $t-$th network uses the information from the past in the sense that it take advantage of the prediction done at the previous step. Nevertheless, this could lead to a fade of the contribution of the first words of the report, while they could be determinant.

To circumvent this issue, one can turn at Long Short Term Memory (LSTM) networks have been introduced by [99]. They can be considered as a particular class of RNN with a design which allows to keep track of the information in a more flexible way. The purpose of LSTM is indeed to introduce a properly designed memory cell whose content evolves with t to keep relevant information through time (and "forget" irrelevant).

As RNN, LSTM are successive blocks of networks. Each block can be described by Figure 4.2. The specificity of LSTM is the presence of a "cell" c_t which differs from the output h_t, and which can

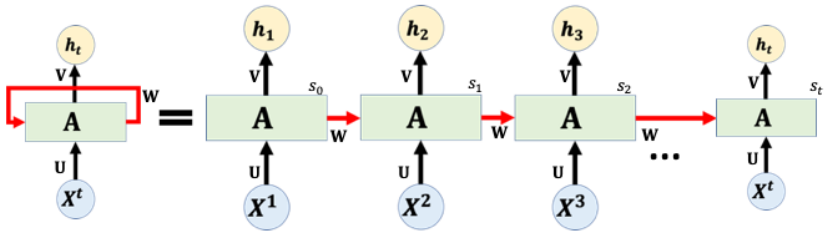


FIGURE 3.4 – A unrolled recurrent neural network and the unfolding representation in time of the computation involved in its forward computation.

be understood as a channel to convey context information that may be relevant to compute the following output h_{t+1} in the next block of the network. The information contained in the cell passes through three gates. These so-called gates are simple operations that permit to keep, add or remove information to the memory cell.

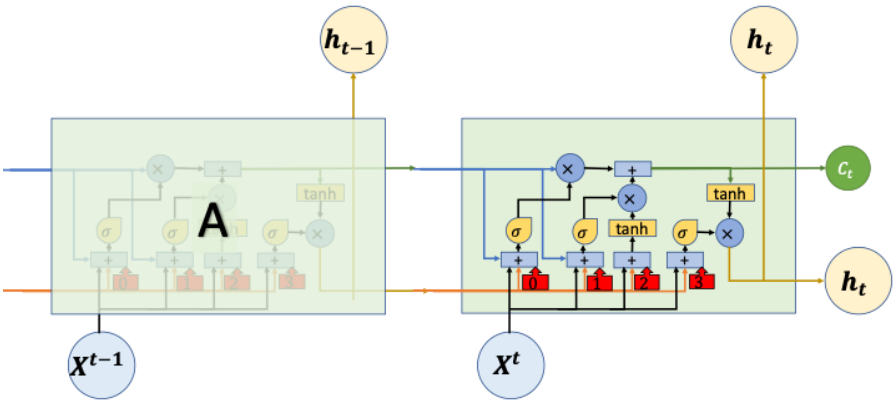


FIGURE 3.5 – Synthetic description of a block of a Long Short Term Memory network.

Following the diagram of Figure 4.2, the memory cell c_t first goes through a multiplicative gate. This means that each component of the vector c_t is multiplied by a number between 0 and 1 (contained in a vector f_t), which allows to keep (value close to 1) or suppress (value close to 0) the information contained in each component of c_t. This vector f_t is computed from the new input x_t and the output of the previous block h_{t-1}.

Next, the modified memory cell enters an additive gate. This gate is used to add information. The added information is a vector $\tilde{C}_t$, again computed from the input x_t and the previous output h_{t-1}. This produces the updated memory cell that passes through the next block.

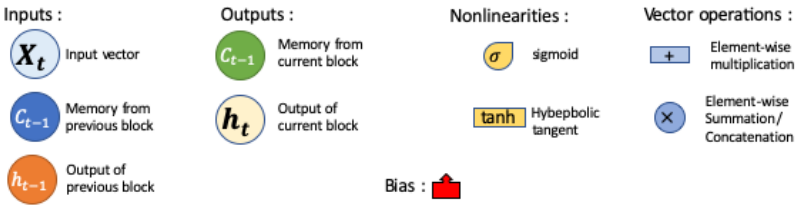


FIGURE 3.6 – LSTM notations.

Moreover, this memory cell is transformed and combined with (x_t, h_{t-1}) through the third step, to produce a new output.

To summarize, the following set of operations are performed in each unit (here, σ denotes the classical sigmoid function, and $[x_t, h_{t-1}]$ the concatenation of the vectors x_t and h_t) :

- Compute $u_t = \sigma(\theta_0[x_t, h_{t-1}]+b_0)$, $v_t = \sigma(\theta_1[x_t, h_{t-1}]+b_1)$, and $w_t = \tanh(\theta_2[x_t, h_{t-1}]+b_2)$.
- The updated value of the memory cell is $c_t = u_t \times c_{t-1} + v_t \times w_t$.
- The new output is $h_t = \sigma(\theta_3[x_t, h_{t-1}] + b_3) \times \tanh(c_t)$.

3.5 Real data analysis

This section is devoted to a real data analysis. The aim is to predict a severity indicator of a claim (which is known only when the claim is fully settled) from expert reports. The main difficulty stands in the fact that the proportion of "severe" claims that require a specific response is quite low. In section 3.5.1, we present the structure of the database and its specificities. Section 3.5.2 is devoted to the inventory of the different network structures and embedding methods that we use. The performance indicators that can be used to assess the quality of the method are shown in section 3.5.3. The results and discussion is made in section 3.5.4. Section 3.5.5 discusses the use of alternative methods for imbalanced text data.

3.5.1 Description of the database

The database we consider gathers 113072 claims from a French insurer, and corresponding expert reports describing the circumstances. These reports have been established at the opening of the claim, and do not take into account counter-expertises. Among these claims, 23% are still open (censored) at the date of extraction. Empirical statistics on the duration of a claim T are shown in Table 3.1.

For the claims that are closed, different severity indicators are known. These severity indicators have been previously computed from the final amount of the claim : $I = 1$ if the claim amount exceeds some quantile of the distribution of the amount, corresponding to the $x\%$ upper part of the distribution. In the situation we consider, this percentage x ranges from 1.5 to 7, which means that we focus on relatively rare events, since $I = 0$ for the vast majority of the claims.

Some elements about the structure of the reports are summarized in Table 3.2 below. To get a first idea of which terms in the report could indicate severity, Table 3.3 shows which are the most represented words, distinguishing between claims with $I = 1$ and claims with $I = 0$.

Categories	min	mean	var	median	max
Standard claims (uncensored)	0	1	1	0,75	16.3
Extreme claims (uncensored)	0,25	3.83	6,93	3,08	16.3
Standard claims (after KM)	0	1.25	2.26	0,83	16.3
Extreme claims (after KM)	0,25	5.24	11.7	4.17	16.3

TABLE 3.1 – Empirical statistics on the variable T, before and after correction by Kaplan-Meier weights ("after KM"). The category "Extreme claims" corresponds to the situation where $I = 1$ for $x = 3\%$ of the claims, while "Standard claims" refers to the 97% lower part of the distribution of the final amount.

Average sentence length	10 words for a standard claim (12 for an extreme one)
Max sentence length	26
Min sentence length	3
Vocabulary size	9749 (without n-grams) 256020 (with n-grams)
Train set size	73684
Test set size	20971
Validation set size	18417

TABLE 3.2 – Summarized characteristics of the reports. The dataset is split into the train set (containing the observations used to fit the parameter of the network), the validation set (used to tune the hyper-parameters), and the test set (used to measure the quality of the model). The vocabulary size "with $n-$grams" is the total of $1-$grams, $2-$grams and $3-$grams.

In both cases, the two most frequent words are "insurer" and "third party", which have to be linked with the guarantee involved. For extreme claims, we notice a frequent presence of a term related to a victim, for example motorcycle, or pedestrian. In addition there are also words related to the severity of the condition of a victim, such as injured or deceased. Moreover, two verbs are present in our top, "to ram" (we used this translation of the French verb "percuter") and "to hit" (in French, "frapper"), which are related to some kind of violence of the event. On the other hand, for "standard" claims, the most frequent words are related to the car. Hence, reports on extreme claims use the lexical field of bodily damage, while the others use the terminology related to material damage.

Let us give two examples of phrases for each category (standard claim and severe claim) :

- standard claim :
  1. "pile-up on the road, 58 vehicles involved."
  2. "insured gets hit at the rear, shock to the rear bumper."
- extreme claim :
  3. "insured hits the solid ground of a roundabout and dies during the transfer to emergency."
  4. "third party does not stop at a stop sign and hits the insured in the front right, air bag triggered."

As one can see from these example, one of the specificity is that reports are often written in a short way, with a syntax which can sometimes be very brief. The vocabulary is also very specific to car insurance. On the other hand, in the third example, one can see that the conclusion that the claim is severe is obvious, since someone died. But example 4 is less obvious. A shock in the front is reported, but without explicit reference to the state of the victims, identification of such a claim as severe is expected to be much harder.

Rank	Extreme	Normal
1	insurer 90%	insurer 87%
2	third party 56%	third party 61%
3	injured 38%	front 46%
4	to ram 30%	way 41%
5	to hit 24%	backside 40%
6	motorcycle 18%	left 20%
7	driver 17%	right 18%
8	pedestrian 16%	side 17%
9	inverse 15%	to shock 14%
10	deceased 13%	control 10%

TABLE 3.3 – Ranking of the words (translated from French) used in the reports, depending on the category of claims (Extreme corresponds to $I = 1$ and Standard to $I = 0$.)

3.5.2 Hyperparameters of the networks and type of embedding

Let us recall that our procedure is decomposed into three steps :

- A preliminary treatment to correct censoring (Kaplan-Meier weighting) ;
- Embedding (determination of some appropriate metric between words) ;
- Prediction, using the embedded words as inputs of a neural network. Depending on the embedding strategy, the prediction phase is either disconnected from the embedding (static method) or done at the same time (after a FastText initialization, non-static method, or a random initialization, random method).

For the last phase, we use LSTM and CNN described in section 3.4. For each of these architectures, we compare the different variations (static, non-static and random). The static and non-static classes of method are expected to behave better, taking advantage of a pre-training via FastText. For the LSTM networks, we consider two cases : when the words are decomposed into $n-$grams ($n = 1, 2, 3$), or without considering $n-$grams. In each case, the idea is to measure the influence of the different rebalancing strategies.

The weights in each of the networks we consider are optimized using the Adagrad optimizer, see [61]. Training is done through stochastic gradient descent over shuffled mini-batches with the Adagrad update rule. The architecture and hyper-parameters of the LSTM and CNN are shown in Table 3.4 and Figure 3.7 and 3.8 below. A ridge regularization is applied to the layer kernel of the output layer, that is introducing a L^2-penalty multiplied by a penalization coefficient α. These parameters are used in every CNN/LSTM model that is considered. The codes are available on

`https://github.com/isaaccs/Insurance-reports-through-deep-neural-networks`.

	CNN	LSTM
dense size 1	400	500
dense size 2	250	300
Ridge coefficient α	0.001	0.004
NB filter	1024	not for LSTM
filtersize	[1,2]	not for LSTM
nb units	not for CNN	[256,128]

TABLE 3.4 – Hyper-parameters of the recurrent neural network used in the real data analysis.

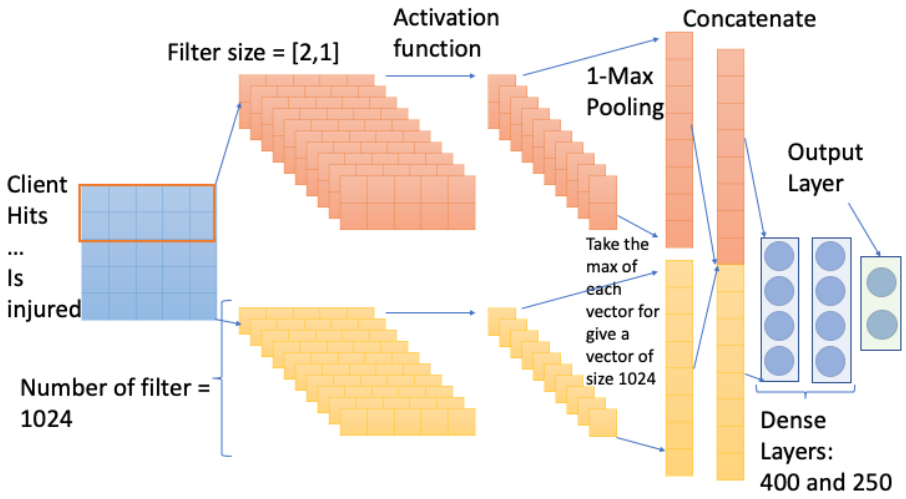


FIGURE 3.7 – Representation of the CNN network used in the real data analysis.

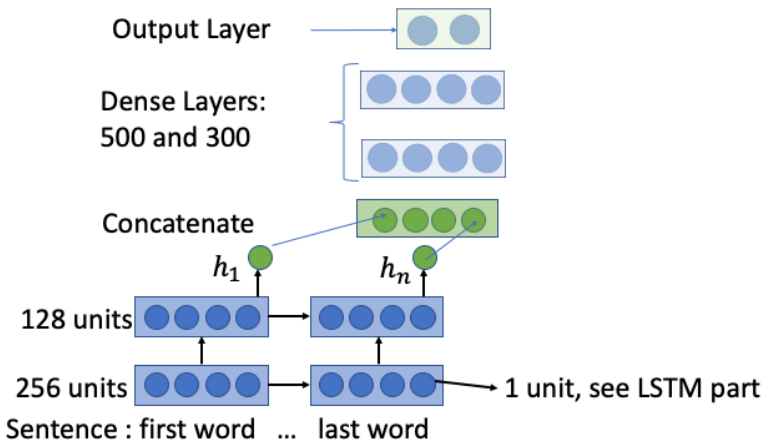


FIGURE 3.8 – Representation of the LSTM network used in the real data analysis. The output of the network is followed by a multilayer perceptron.

Our model is implemented using Tensorflow, see [2], on Python for deep learning models, and we use sk-learn, see [160], for the Machine Learning models.

3.5.3 Performance indicators

In this section, we describe the criteria that are used to compare the final results of our models. Due to the small proportion of observations such that $I = 1$, measuring the performance only by the well-predicted responses would not be adequate, since a model which would systematically predict 0 would be ensured to obtain an almost perfect score according to this criterion. Let us introduce some notations :

- TN is the number of negative examples ($I = 0$) correctly classified, that is "True Negatives".
- FP is the number of negative examples incorrectly classified as positive ($I = 1$), that is "False Positives".
- FN is the number of positive examples incorrectly classified as negative ("False Negatives").
- TP is the number of positive examples correctly classified ("True Positives").

We then define Recall and Precision as,

$$\begin{aligned} \text{Recall} &= \frac{TP}{TP+FN}, \\ \text{Precision} &= \frac{TP}{TP+FP}. \end{aligned}$$

So Recall is the proportion of 1 that have been correctly predicted with respect to the total number of 1. For Precision, the number of correct predictions of 1 is compared to the total of 1-predicted observations. F_1-score (see for example [4]) is a way to combine these two measures, introducing

$$F_1 = \frac{2 \times Recall \times Precision}{Recall + Precision}.$$

The F_1-score conveys the balance between the precision and recall.

Remark :

F_1-score is a particular case of a more general family of measures defined as

$$F_\beta = \frac{(1+\beta^2) \times Recall \times Precision}{Recall + \beta^2 \times Precision},$$

where β is a coefficient to adjust the relative importance of precision versus recall, decreasing β leads a reduction of precision importance. In order to adapt the measure to cost-sensitive learning methods [39] proposed to rely on

$$\beta = \frac{c(1,0)}{c(0,1)}, \tag{3.6}$$

where $c(1,0)$ (resp. $c(0,1)$) is the cost associated to the prediction of a False Negative (resp. of a False Positive). The rationale behind the introduction of such F_β measure is that misclassifying within the minority class is typically more expansive than within the majority class. Hence, considering the definition (3.6) of β, improving Recall will more affect F_β than improving Precision. In our practical situation, it was difficult to define a legitimate cost for such bad predictions, which explains why we only considered F_1-score among the family of $F_\beta-$measures.

3.5.4 Results

As we already mention, $I = 1$ corresponds to a severe event, that is the loss associated with the claim corresponds to the $x\%$ upper part of the loss distribution. We made this proportion vary from

1.5% to 7% to see the sensitivity to this parameter. Table 3.5 shows the benefits of using the different resampling algorithms of section 3.2.3 in the case where the proportion of 1 in the sample is 3% (which is an intermediate scenario considering the range of values for x that we consider). The neural networks methods are benchmarked with a classical logistic predictor, and two other competing machine learning methodology (Gradient Boosting and Random Forests, see [74]). To make think comparable, all these alternatives methodologies are combined with the FastText embedding (as in the static methods). Moreover, we apply the same resampling algorithms of section 3.2.3 to these methods. For Algorithm 6, we considered a balanced case, where the proportion of 1 in the bootstrap samples is 50%, and a "lightly" rebalanced case where these proportion is only 10%. This choice of 10% was motivated by Table 1 in [209] : it corresponds to a situation that the authors qualify as "moderately imbalanced", instead of "extremely imbalanced". In Table 3.6, confidence intervals are provided for the performance indicators.

Let us first observe that the network methodologies lead to a relatively good precision if one does not use rebalancing strategies. This is due to the fact that precision only measures the proportion of true positive among all the claims that have been predicted to be 1. Typically, the networks methodologies, in this case, predict correctly the "obviously" severe claims, but miss lots of these severe claims. This explains why recall and F_1-score are much lower.

The rebalancing algorithm mostly benefit the machine learning methodologies. One also observes that the embedding methodologies based on FastText (static and non-static methods) leads to better performances. The performance of all techniques in terms of F_1-score is not spectacular (in the sense that it is not close to 1), but this has to be related with the complexity of the problem (predicting a relatively rare class of events). Compared to the basic logistic method (after embedding), the best network (LSTM without considering n grams, with the lightly balanced bagging algorithm) leads to an improvement of 20% in terms of F_1-score.

Let us also observe that the LSTM network architecture which does not rely on $n-$grams behaves better than the LSTM which actually uses these $n-$grams. This may be counterintuitive, since the use of $n-$grams is motivated by the idea that one could use them to take the context of the use of a word into account. Nevertheless, considering $n-$grams increases the number of parameters of the network, while context information is already present in the embedding methodologies. On the other hand, CNN methods have lower performance than LSTM ones, but the performances stay in the same range (with a computation time approximatively 4 times smaller to fit the parameters of the network).

Method	Model	type Embedding	precision	recall	F1-score
Classical	Logistic	static	0.26	0.30	0.28
	Random Forest	static	0.20	0.22	0.21
	Gradient Boosting	static	0.17	0.31	0.22
	CNN	rand	0.61	0.16	0.26
		static	0.68	0.08	0.14
		non-static	0.78	0.06	0.12
	LSTM no n-Grams	rand	0.78	0.06	0.11
		static	0.58	0.14	0.22
		non-static	0.66	0.11	0.19
	LSTM n-Grams	rand	0.72	0.08	0.14
		static	0.57	0.13	0.21
		non-static	0.57	0.14	0.22
Balanced	Logistic	static	0.34	0.34	0.34
	Random Forest	static	0.25	0.40	0.31
	Gradient Boosting	static	0.22	0.34	0.27
	CNN	rand	0.28	0.36	0.30
		static	0.28	0.35	0.31
		non-static	0.28	0.48	0.33
	LSTM no n-Grams	rand	0.34	0.31	0.32
		static	0.37	0.37	0.37
		non-static	0.28	0.46	0.35
	LSTM n-Grams	rand	0.25	0.27	0.26
		static	0.31	0.46	0.37
		non-static	0.29	0.41	0.34
Randomly	Logistic	static	0.30	0.38	0.34
	Random Forest	static	0.22	0.43	0.29
	Gradient Boosting	static	0.25	0.37	0.30
	CNN	rand	0.07	0.30	0.12
		static	0.32	0.45	0.37
		non-static	0.33	0.42	0.37
	LSTM no n-Grams	rand	0.33	0.33	0.33
		static	0.09	0.63	0.16
		non-static	0.34	0.48	0.40
	LSTM n-Grams	rand	0.24	0.29	0.27
		static	0.23	0.53	0.32
		non-static	0.29	0.39	0.33
Lightly	Logistic	static	0.34	0.39	0.36
	Random Forest	static	0.27	0.38	0.31
	Gradient Boosting	static	0.25	0.33	0.29
	CNN	rand	0.43	0.37	0.40
		static	0.39	0.44	0.41
		non-static	0.41	0.44	0.42
	LSTM no n-Grams	rand	0.42	0.33	0.37
		static	0.40	0.45	0.42
		non-static	0.47	0.40	0.43
	LSTM n-Grams	rand	0.33	0.33	0.33
		static	0.31	0.46	0.37
		non-static	0.34	0.41	0.37

TABLE 3.5 – Influence of the constrained bagging algorithms of section 3.2.3. For "Classical," no rebalancing algorithm has been used. "Balanced" corresponds to algorithm 5 with an equal proportion of 1 and 0 in each sample. "Randomly" corresponds to algorithm 6 with an equal proportion (in average) of 1 and 0 in each sample. "Lightly" corresponds to algorithm 6 with 10% of 1 (in average) in each sample. In the Random Forest algorithm, we used 200 trees, with a Gini criterion and a maximal depth of 3. In the Gradient Boosting algorithm, 200 trees were used, and a deviance loss with a learning rate of 0.1.

Method	Model	type Embedding	CI precision	CI recall	precision CI for bagging estimator	recall CI for bagging estimator	F1 CI for bagging estimator
Classical	Logistic	static	(0.976, 0.981)	(0.975, 0.98)	None	None	None
	Random Forest	static	(0.946, 0.953)	(0.977, 0.981)	None	None	None
	Gradient Boosting	static	(0.965, 0.971)	(0.974, 0.978)	None	None	None
	CNN	rand	((0.99, 0.993)	(0.972, 0.977)	None	None	None
		static	(0.998, 0.999)	(0.97, 0.974)	None	None	None
		non-static	(0.998, 0.999)	(0.969, 0.974)	None	None	None
	LSTM no n-Grams	rand	(0.998, 0.999)	(0.968, 0.973)	None	None	None
		static	(0.996, 0.998)	(0.971, 0.976)	None	None	None
		non-static	(0.996, 0.998)	(0.97, 0.975)	None	None	None
	LSTM n-Grams	rand	(0.997, 0.999)	(0.969, 0.974)	None	None	None
		static	(0.995, 0.997)	(0.971, 0.976)	None	None	None
		non-static	(0.996, 0.998)	(0.971, 0.976)	None	None	None
Balanced	Logistic	static	(0.901, 0.911)	(0.968, 0.973)	(0.33 , 0.35)	(0.29, 0.34)	(0.33 , 0.34)
	Random Forest	static	(0.925, 0.933)	(0.959, 0.965)	(0.25, 0.27)	(0.34, 0.40)	(0.3, 0.32)
	Gradient Boosting	static	(0.939, 0.945)	(0.956, 0.962)	(0.20, 0.24)	(0.31,0.36)	(0.24,0.28)
	CNN	rand	(0.932, 0.94)	(0.959, 0.964)	(0.17,0.37)	(0.22, 0.67)	(0.25, 0.32)
		static	(0.918, 0.926)	(0.962, 0.967)	(0.17,0.29)	(0.34, 0.67)	(0.27, 0.34)
		non-static	(0.945, 0.951)	(0.959, 0.965)	(0.21,0.32)	(0.38, 0.58)	(0.29, 0.36)
	LSTM no n-Grams	rand	(0.923, 0.931)	(0.963, 0.968)	(0.20,0.34)	(0.44, 0.65)	(0.26, 0.35)
		static	(0.953, 0.959)	(0.961, 0.9663)	(0.27,0.46)	(0.04, 0.49)	(0.07, 0.37)
		non-static	(0.945, 0.951)	(0.96, 0.965)	(0.28,0.35)	(0.27, 0.5)	(0.3, 0.38)
	LSTM n-Grams	rand	(0.925, 0.933)	(0.956, 0.962)	(0.15,0.32)	(0.19, 0.44)	(0.2, 0.29)
		static	(0.869, 0.879)	(0.966, 0.972)	(0.22,0.38)	(0.29, 0.68)	(0.33, 0.37)
		non-static	(0.917, 0.925)	(0.964, 0.969)	(0.21,0.3)	(0.06, 0.46)	(0.06, 0.34)
Randomly	Logistic	static	(0.951, 0.957)	(0.96, 0.965)	(0.06, 0.30)	(0.39,0.93)	(0.15 , 0.34)
	Random Forest	static	(0.927, 0.935)	(0.96, 0.966)	(0.06, 0.24)	(0.22, 0.97)	(0.09, 0.29)
	Gradient Boosting	static	(0.869, 0.879)	(0.959, 0.965)	(0.25 , 0.33)	(0.03, 0.37)	(0.10 , 0.31)
	CNN	rand	(0.39, 0.404)	(0.946, 0.956)	(0.0, 0.97)	(0.0, 0.57)	(0.0, 0.19)
		static	(0.917, 0.925)	(0.967, 0.972)	(0.06, 0.42)	(0.16, 0.99)	(0.12, 0.37)
		non-static	(0.93, 0.936)	(0.966, 0.971)	(0.07, 0.33)	(0.42, 1)	(0.11, 0.37)
	LSTM no n-Grams	rand	(0.96, 0.965)	(0.958, 0.964)	(0.0, 0.36)	(0.0, 0.51)	(0.0, 0.36)
		static	(0.951, 0.957)	(0.962, 0.968)	(0.05, 0.38)	(0.08, 0.91)	(0.1, 0.37)
		non-static	(0.939, 0.946)	(0.956, 0.962)	(0.17, 0.41)	(0.24, 0.66)	(0.27, 0.42)
	LSTM n-Grams	rand	(0.932, 0.939)	(0.958, 0.964)	(0.0,0.36)	(0, 0.43)	(0, 0.3)
		static	(0.94, 0.947)	(0.962, 0.968)	(0.0,0.32)	(0, 0.53)	(0, 0.37)
		non-static	(0.79, 0.802)	(0.952, 0.959)	(0.08,0.29)	(0.38, 0.89)	(0.14, 0.35)
Lightly	Logistic	static	(0.945, 0.951)	(0.962, 0.967)	(0.34,0.34)	(0.37 ,0.39)	(0.35, 0.46)
	Random Forest	static	(0.929, 0.936)	(0.959, 0.965)	(0.27, 0.27)	(0.35,0.38)	(0.3,0.31)
	Gradient Boosting	static	(0.877, 0.887)	(0.962, 0.967)	(0.24,0.26)	(0.32, 0.40)	(0.27, 0.30)
	CNN	rand	(0.959, 0.965)	(0.962, 0.967)	(0.34, 0.43)	(0.33,0.43)	(0.34,0.41)
		static	(0.945, 0.952)	(0.965, 0.971)	(0.33, 0.43)	(0.33,0.48)	(0.37,0.41)
		non-static	(0.956, 0.961)	(0.964, 0.967)	(0.36, 0.41)	(0.35,0.49)	(0.37,0.42)
	LSTM no n-Grams	rand	(0.948, 0.954)	(0.961, 0.966)	(0.35, 0.45)	(0.26,0.34)	(0.32,0.37)
		static	(0.95, 0.955)	(0.963, 0.962)	(0.36, 0.41)	(0.36,0.46)	(0.38,0.42)
		non-static	(0.97, 0.974)	(0.96, 0.965)	(0.39, 0.47)	(0.33,0.43)	(0.39,0.43)
	LSTM n-Grams	rand	(0.958, 0.963)	(0.956, 0.962)	(0.17,0.33)	(0.24,0.33)	(0.23,0.33)
		static	(0.956, 0.962)	(0.961, 0.967)	(0.33,0.42)	(0.30,0.46)	(0.32,0.38)
		non-static	(0.939, 0.946)	(0.964, 0.969)	(0.29,0.36)	(0.31,0.49)	(0.33,0.39)

TABLE 3.6 – Confidence Interval (CI) of the scoring. The CI for Precision and Recall (columns 4 and 5) are computed using a beta distribution following the methodology of [87]. For bagging (columns 6 to 8), the CI is based on percentile bootstrap following [97].

Table 3.7 shows the influence of the proportion x on the performance (we only report a selected number of methods for the sake of brevity. We observe that the performance of the logistic method decreases with the proportion of 1 contained in the sample. On the other hand, the network-based methods' performance stay relatively stable.

Extreme Values Percent	Model	type Embedding	precision	recall	f1-score
1.5%	Logistic	static	0.19	0.24	0.21
	CNN	rand	0.38	0.41	0.39
		static	0.49	0.36	0.41
		non-static	0.40	0.42	0.41
	LSTM	rand	0.42	0.33	0.37
		static	0.34	0.47	0.39
		non-static	0.47	0.40	0.43
3.5%	Logistic	static	0.27	0.35	0.30
	CNN	rand	0.39	0.47	0.43
		static	0.37	0.48	0.42
		non-static	0.35	0.45	0.40
	LSTM	rand	0.33	0.37	0.34
		static	0.35	0.49	0.40
		non-static	0.34	0.49	0.40
5%	Logistic	static	0.25	0.42	0.31
	CNN	rand	0.41	0.44	0.42
		static	0.40	0.46	0.43
		non-static	0.42	0.45	0.44
	LSTM	rand	0.22	0.31	0.26
		static	0.37	0.49	0.42
		non-static	0.43	0.37	0.40
7%	Logistic	static	0.29	0.45	0.35
	CNN	rand	0.40	0.47	0.43
		static	0.43	0.46	0.44
		non-static	0.38	0.46	0.41
	LSTM	rand	0.32	0.34	0.33
		static	0.42	0.41	0.42
		non-static	0.43	0.40	0.41

TABLE 3.7 – Performance of the different methodologies (combined with the different embedding technics). The first column indicates the percentage of 1 in the sample.

3.5.5 Alternative methods for text analysis of imbalanced data

Among the techniques that are frequently used when it comes to dealing with imbalanced data, SMOTE, see [39], has not been used in our context, because this approach is designed for image and not adapted to text analysis. ForesTexter, see [214], is an algorithm based on random forests which can be used for text categorization, with promising results. Therefore, we compared the results of our approach with ForesTexter. Through this method, we obtained a Precision of 0.58, a Recall of 0.07, and a F1-Score of 0.12 (when the percentage of 1 is 3%).

Hence, the Precision is much better for ForesTexter. This means that, when ForesTexter predicts an extreme claims, it tends to be right (at least, more than with our procedure). On the other hand, it has a poor performance in terms of Recall, which means it tends to "under-detect" these extreme claims. In other words, this approach tends to only detect obvious extreme claims. For insurance applications, the F1-Score seems a much adapted metric.

3.6 Conclusion

In this paper, we proposed a detailed methodology to perform automatic analysis of text reports in insurance, in view of predicting a rare event. In our case, the rare event is a particularly severe claim. This question of clustering such claim is of strategic importance, since it allows to operate a particular treatment of the claims that are identified as "extreme". Four steps of our method are essential :

- correction of censoring via survival analysis techniques ;
- compensation of the rarity of the severe events via rebalancing bagging techniques ;

— a proper representation of the words contained in the reports via an appropriate embedding technique ;
— the choice of a proper neural network architecture.

Long Short Term Memory networks, associated with a performant embedding method, appear to be promising tools to perform this task. Nevertheless, learning rare events is still a hard task, especially in such insurance problem where the volume of data is limited, especially when dealing with such a specific vocabulary. Insurance is, of course, not the only sector where specific terminologies can influence the behavior of such techniques, see for example [131] for similar biomedical problematics. The integration of additional variables to increase the information on the claims should be essential to improve the prediction. Moreover, let us mention that, in this work, we only considered information available at the opening of the claim to predict its outcome. For long development branches, the incorporation of updated information on the evolution of the claims could be determinant and should be incorporated in the methodology.

Chapitre 4

Prediction of the outcome of insurance claims with deep neural networks

In this paper, we develop a methodology to perform the prediction on an open insurance claim (RBNS, Reported But Not Settled) from a set of complex covariates with various structures (structured and unstructured data). The technique combines different deep neural networks architectures (such as Long Short Term Memory for text data) with survival analysis prediction methods (to predict the time of settlement of the claim). The deep learning methods are used to extract features from our complex data, hence to perform dimension reduction. These features may be plugged in a final neural network predictor, or combined with more intelligible models like a Generalized Linear Model, if the need for interpretation is more important than the quality of the prediction. A real data analysis illustrates the technique.

4.1 Introduction

For many third-party insurance guarantees, the settlement process of claims may be quite long, and the uncertainty about their final amount can be huge. Accurately predicting their final outcome is then an important issue. It is, of course, a matter of improving the vision one has of the amount of reserves that should be gathered. More generally, it is a question of risk management since it can give indications on actions to be performed before the settlement of the claim to try to reduce its impact. The increase of the volume of available data and the blooming of machine learning methodologies prompts to develop these techniques in the field of claim analysis. The present paper presents a general technique that can be used to performed this task, in a context where the available information is complex and of various types (unstructured data as text, quantitative variables, evolving information...). The core of the methodology is the combination of several deep neural networks architectures, with a particular attention on the treatment of the prediction of the time of settlement of a claim.

In the recent literature, machine learning techniques have been considered as promising tools for reserving purpose. Individual reserving, see for example [152], [154] or [164], aims to use all available information on a RBNS (Reported But Not Settled) claim to determine its final amount. In this perspective, [138] or [137] used regression trees to perform this task. Other methodologies, like boosting (see [63]) or random forests (see [17]) have also been proposed, while [149] proposed neural network methodologies. Neural network methodologies have also been combined successfully with traditional "triangle" reserving methods, see [217], [216] or [122].

The performance of neural networks is strongly linked to the determination of an architecture appropriate for the problem they aim to solve. From the generic multilayer perceptron (see [176], [155]), deep learning structures have been successively used to handle more complex types of data, like Convolutional Neural Networks (see for example [125], [188], [111]) in image analysis, or Recurrent Neural Networks (RNN, see [145], [148]) in dynamic problems such that text analysis or speech recognition. In this paper, we turn to a particular type of RNN, that is Long Short Term Memory (LSTM) networks (see [99], [183] or [42]) which gave particularly promising results in text analysis by their ability to keep track of important informations in a text, even if they are separated by an important number of words. In fact, LSTM aim to mimic the concept of memory by introducing an appropriate mechanism in their structure that allows them to select information.

The reason for turning to such elaborate networks is the need to automatically treat text reports coming from several steps in the life of the claim (declaration of claims, expert reports...). Providing an automatic analysis of these elements is strategic, since it provides a fast detection of problematic claims, and potentially a reduction of the cost (and sometimes the imprecisions or collusions) of experts by identifying which claims require more attention. [182] proposed a way to use LSTM networks to identify extremely severe claims, based only on text reports available at their opening, with the difficulty that learning methods' performance is weaker when focusing on rare events.

In the present paper, we propose a general method that does not only apply to claims just after their occurrence, but may be updated at each stage of the (potentially long) settlement process. Our methodologies separates between variables of different types, each of them receiving a particular treatment. For example, the total settlement time is a variable which is usually strongly correlated with the final amount of the claim. Survival analysis techniques are used to handle this variable and diminish the impact of censoring, as in [137], [136] or [182]. Deep learning methods are used to perform dimension reduction in this huge amount of covariates, extracting features that are used in a final prediction model. This last model may or may not be a "black box" : if intelligibility is required for operational purpose, a Generalized Linear Model can for example be used in the end (up to a reduction of the prediction quality compared to fully machine learning based techniques), based on the features determined by our deep architecture.

The rest of the paper is organized as follows. In section 4.2, we explain the structure of our data and the main steps of our approach. Section 4.3 gives a more detailed description of the neural network and survival analysis technique that we use. A particular focus on deep neural architecture such as LSTM is done in section 4.4. Finally a data analysis illustrates the performance on our method on a real case.

4.2 Framework and general methodology

This section is devoted to the description of the structure of our data. In section 4.2.1, we define the different types of variables that are used to build a prediction model for the final amount of an open claim. Our methodologies can be decomposed into two main steps, explained in section 4.2.2, which separates a general prediction model from a prediction of the time before settlement of the claim.

4.2.1 Data

In our framework, a claim is described by the set of variables $(A, T, \mathbf{X^{(1)}}, \mathbf{X}^{(2)}, \mathbf{Z})$. A denotes the final amount of the claim, which is the quantity we wish to predict. T is the time before settlement

of the claim. This variable is expected to be strongly correlated with A : a long development time indicates difficulties in solving the problems raised by the claim, hence it tends to be linked with a large final amount. On the other hand, this variable is only known at the resolution of the claim, that is when A is known. This explains why, among the covariates, T requires a particular treatment, as it will be explained below.

The rest of the covariates are decomposed into three types of variables. $\mathbf{X}^{(1)}$ denotes a text report describing the claim. This text report has been produced at the opening. It is high-dimensional (it can be seen as a matrix, see section 4.4.2, where the encoding of $\mathbf{X}^{(1)}$ is described). $\mathbf{X}^{(2)}$ gathers updates on the status of the claim through its "life". It consists in qualitative and quantitative variables describing the events (expertises, counter-expertises...) during the development. It can be understood as a multivariate time-series. Finally, $\mathbf{Z}$ contains fixed covariates (age of the victim, profession...) that are known at the opening of the claim and do not change through time.

To calibrate our model, our observations are made of a database of both closed and open claims. This aspect is crucial : open claims only convey a partial information on the phenomenon (the final amount is not known), but if the calibration of our models were to be performed only on closed claims, this would introduce some selection bias. As already mention, a positive (strong) correlation between the time of settlement T and the amount A is suspected. Considering closed claims only would lead to a database where claims with small T (hence, in average, a small A) would be over-represented. This could lead to an under-estimation of the typical values taken by A, as pointed out by [136] or [137]. This has to be linked with the phenomenon of censoring in survival analysis, see [70].

Hence, let us introduce

$$\begin{cases} Y &= \inf(T, C), \\ \delta &= \mathbf{1}_{T \leq C}, \end{cases}$$

where C is a censoring variable (that is C is the time after which the information on the claim is lost, either because there has been a retrocession of a portfolio, or because the extraction of the database has been done before the conclusion of the claim settlement process). In the following, we assume that C is independent from the vector $(A, T, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{Z})$.

Our observations are considered as a set of i.i.d. replications $(\delta_i A_i, \delta_i, Y_i, \mathbf{X}_i^{(1)}, \mathbf{X}_i^{(2)}, \mathbf{Z}_i)_{1 \leq i \leq n}$ of the random vector $(\delta A, \delta, Y, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{Z})$, where n is the total number of claims contained in our database.

4.2.2 A two steps procedure

Our procedure to predict A can be decomposed in two steps :

- estimate $a(t, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{z}) = E[A|T = t, \mathbf{X}^{(1)} = \mathbf{x}^{(1)}, \mathbf{X}^{(2)} = \mathbf{x}^{(1)}, \mathbf{Z} = \mathbf{z}]$ (which is the best possible predictor of A from a quadratic loss perspective, knowing $(t, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{z})$) by a model $\widehat{a}$;
- since T is unknown for an open claim, the second step is to compute a predictor $\widehat{T}$ of T from the available covariates.

Then, the two pieces are joint together, the final prediction of the amount being

$$\widehat{A} = \widehat{a}(\widehat{T}, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{Z}).$$

Via the first step, we take into account the influence of T on the amount, avoiding a model that would only be based on the informations available at the current state of the evolution of an open claim.

These two parts of the works are not done the same way : the idea is to develop a very accurate model to compute function $\widehat{a}$, which represents our understanding of the impact of the different variables on our response. On the other hand, we choose to rely on a rougher model for the prediction of T, which is a less volatile variable than A. Classical survival analysis methods will be used to predict this variable, while deep neural networks methodologies are used for the estimation of a.

The structure of such models is described in section 4.3 below.

4.3 Structure of the prediction models

This section described the procedure we use to develop a prediction model for A (sections 4.3.1 to 4.3.3), before finally turn to the question of predicting the time before settlement in section 4.3.4. Regarding the prediction of A, we first start by decomposing A into a small number of classes, in order to simplify the prediction problem. This is done in section 4.3.1. We then introduce in section 4.3.2 the principle of a neural network methodology which has two phases : feature extraction, and final prediction step. The typical models that can be used for this final step are described in section 4.3.3.

4.3.1 Decomposition of the variable A into two modalities

As already mentioned, the amount A may be highly volatile. To simplify its study, we decompose A into two classes, separating "severe" claims from standard ones :

- if $0 \leq A < a_0$, it is considered as a standard claim, where a_0 is some operational limit under which it is considered that no particular attention should be paid to this case ;
- if A exceeds a_0, then the claim is considered as particularly severe.

Hence, we define a variable I which takes the value 1, 2 depending that A belongs to the first or the second category. Of course, in full generality, additional categories could be introduced. This decomposition into two classes has also to be linked with some statistical properties of the variable A : A has a continuous distribution, which for highly volatile guarantees, can be considered as heavy-tailed. In this case, a_0 can be understood as a threshold beyond which A can be approximated by a Generalized Pareto Distribution (see section 4.5.2), while the distribution of A when $I = 1$ is more concentrated.

It is of course easier to predict the value of I, which has only few labels, from the complex structure of our covariates than directly predict A. Then, from our data, we compute

$$\widehat{m}_i = \frac{1}{\sum_{j=1}^{n} \mathbf{1}_{I_j=i}} \sum_{k=1}^{n} A_k \mathbf{1}_{I_k=i},$$

which is an estimator of $m_i = E[A|I = i]$. Based on a prediction $\widehat{I}$ of I, we define $\widehat{A} = \widehat{m}_I \widehat{I}$. The next two sections are devoted to the construction of this predictor $\widehat{I}$.

Remark. *Here, to simplify, we only focus on the mean value of A. But, to compute reserves our to simply have a better understanding of the distribution of A, one may also fit a distribution to A for $I = i$ for $i = 1, 2$.*

4.3.2 Feature extraction and computation of $\widehat{a}$

As we already mentioned, the structure of our covariates used to predict I (and then A) is quite complex. Text data $\mathbf{X}^{(1)}$ requires a particular treatment, while $\mathbf{X}^{(2)}$ is highly dimensional. Our procedure consists in performing a significant dimension reduction of the vector $(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{Z})$, treating each type of variable separately in order to take its particular nature into account. Next, this set of features variables is joint with the covariate T and sent as input of a regression method.

Since we want to use the variable T in our prediction, our model is built on the set of closed claims only (that is such that $\delta_i = 0$). Nevertheless, as we already mentioned, in this database there is an under-representation of the large claims. To correct this selection bias, we first modify the database by duplicate the large claims to compensate this under-representation. The information contained in the open claims is used to determine how many times an observation should be replicated. Such an algorithm has already been studied in [182] and is similar to the weighting strategy developed by [220] to compensate censoring. For the sake of exhaustivity, it is summarized in the Appendix, section D.0.1.

The procedure can be summarized as follows :

1. Compute three separate predictors $\tilde{I}_1 = \tilde{I}(\mathbf{X}^{(1)})$, $\tilde{I}_2 = \tilde{I}(\mathbf{X}^{(2)})$, $\tilde{I}_3 = \tilde{I}(\mathbf{Z})$, taking as inputs only $\mathbf{X}^{(1)}$ for $\tilde{I}_1$, only $\mathbf{X}^{(2)}$ for $\tilde{I}_2$, only $\mathbf{Z}$ for $\tilde{I}_3$. For $\tilde{I}_1$ and $\tilde{I}_2$, complex deep neural networks structures are used, namely Long Short Term Memory networks (described in section 4.4.3), combined with an text embedding step (see section 4.4.2) for the text data. For $\tilde{I}_3$ a standard neural network (multilayer perceptron, see section 4.4.1) is required.
2. From the neural networks used to compute $\tilde{I}_1$ to $\tilde{I}_3$, extract features $\mathbf{F}_1$ to $\mathbf{F}_3$ (these features are made of the last layer of the networks, see section 4.4.4 for more details).
3. Fit a prediction model of A from $(\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3, T)$.

For this last step, any regression/prediction model can be used, from a classical parametric model such as Generalized Linear Model (GLM) to more recent machine learning methods. A neural network can be used as in the first step, but there is no obligation. In section 4.3.3, we briefly list a few number of categories of predictors that can be plugged into this last step.

4.3.3 Final predictor

In this section, we use the synthetic notation $\mathbf{F} = (\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3, T)$ to describe our covariates obtained after feature extraction.

4.3.3.1 Generalized Linear Models

Generalized Linear Models, see [142], consists in assuming that

$$g(E[I|\mathbf{F} = \mathbf{f}]) = \beta_0 + \beta_1^{tr}\mathbf{f},$$

where g is a known monotonous link function and (β_0, β_1) are finite dimensional parameters (here β_1^{tr} denotes the transpose vector of β_1). Moreover, the conditional distribution of I conditionally to $\mathbf{F} = \mathbf{f}$ belongs to an exponential family of distributions, which is used to compute a log-likelihood and fit the parameters.

4.3.3.2 Regression Trees and Random Forests

Regression Trees have been introduced by [33]. Compared to GLM, they allow to introduce more nonlinearities in the final model. A tree-based structure allows to determine a function of the following form,

$$r(\mathbf{f}) = \sum_{i=1}^{K} r_i \mathbf{1}_{\mathbf{f} \in R_i}.$$

Here $(R_i)_{1 \leq i \leq K}$ are called "rules", that is a partition of the space of the covariates $\mathbf{F}$ into disjoints classes, which correspond to the leaves of the tree produced by the CART (Clustering And Regression Tree) algorithm. Random forests, see [32], can be seen as aggregations of small regression trees, each of them only fitted from a small randomly selected number of covariates and from a different bootstrap sample of the original data. Compared to regression trees, forests provide more stable predictions : the structure of the tree, hence the set of rules, may change significantly if new observations are introduced in the database. Nevertheless, random forests, as neural networks, are subject to a kind of "black-box" effect, making them more difficult to interpret than regression trees.

4.3.3.3 Boosting methods

Boosting methods, like in XGBoost, see [41] or [40], are iteratively improved methods which first start with poor predictors, and then ameliorate them by investigating through a gradient boosting algorithm. Typically, the gradient of the loss at the current state of estimation is computed, to identify the direction following which the prediction has to be improved. A survey on boosting methods can be found in [116]. Many applications to insurance have been proposed recently, see [68], [57] or [162].

4.3.4 Prediction of T

For the prediction of T, we rely on a simple survival analysis model. Cox regression model, see [51], consists in assuming that the conditional survival function of T, say $S(t|\mathbf{z}) = \mathbb{P}(T \geq t|\mathbf{Z} = \mathbf{z})$ derives from a baseline function S_0 in the following way,

$$S(t|\mathbf{z}) = S_0(t)^{\exp(\gamma_0^{tr}\mathbf{z})}.$$

In this semiparametric model, both parameter γ_0 and baseline function S_0 are unknown. The parameter γ_0 is estimated (from the all dataset, that is the dataset containing both open and closed claims) maximizing Cox pseudo-likelihood, see [70]. The Breslow estimator, see again [70], can then be used to estimate the baseline.

Let us consider a claim that is open since a duration of t (that is $Y = t, \delta = 0$), and for which $\mathbf{Z} = \mathbf{z}$. Based on the two estimators $\widehat{S}_0$ and $\widehat{\gamma}_0$, we compute the following predictor of T,

$$\widehat{T} = \frac{\int_t^\infty \widehat{S}_0(u)^{\exp(\widehat{\gamma}_0^{tr}\mathbf{z})} du}{\widehat{S}_0(t)^{\exp(\widehat{\gamma}_0^{tr}\mathbf{z})}}, \tag{4.1}$$

which is a consistent estimator (under the Cox regression assumption) of the quantity $E[T|\mathbf{Z} = \mathbf{z}, T \geq t]$. Let us observe that the integral in (4.1) can be computed without any approximation, since the Breslow estimator $\widehat{S}_0$ produces a piecewise constant function.

Alternative models to Cox Regression could also be considered : fully parametric models, such as Accelerated Failure Time Models, see [211], or other semiparametric regression models such that, for example, [199], [135].

4.4 Neural networks and Long Short Term Memory

This section is devoted to a detailed description of the neural network part of our procedure. The general terminology of neural networks is considered in section 4.4.1. Since we are dealing with text data, the question of a proper and efficient representation of the reports we use as inputs is crucial. We briefly describe in section 4.4.2 the concept of embedding which is essential to this representation. The Long Short Term Memory architecture, which is a particular deep learning architecture adapted to text data, is described in section 4.4.3. Finally, in section 4.4.4, we give precisions on how features can be extracted from these networks, as mentioned in the description of our procedure (section 4.3.2).

4.4.1 Neural networks terminology and notations

A neural network is an over-parametrized model that aims to predict a response variable. By over-parametrized, we mean that the number of parameters (or weights, using the classical neural network terminology) is huge, potentially larger than the number of observations. The most classical type of neural networks, the multilayer perceptron see [155], is now widely used as a regression tool, and has been applied to many questions in insurance like, for example, non-life insurance pricing as in [218], or longevity analysis, see [92], or reserving, see [217].

The covariates, also called inputs of the network, are sent in different units (that is neurons) which perform simple operations, as described in Figure 4.1. Typically, a linear combination of the covariate is done (the coefficients are called weights), a bias coefficient is introduced playing the same role as the intercept coefficient in traditional regression, and an activation function is applied to introduced linearity. The first set of neurons which receive directly the information from the covariates is called the first hidden layer, and the output of each neuron is sent in another layer.

The last layer, which can be made of one single neuron or more, is called the output layer, and contains the final prediction of the response by the network. Formally, a neural network is a function $m(\mathbf{x}, \theta)$, where θ belongs to a large dimension space Θ. The value of the parameter θ is iteratively optimized, the aim being to minimize a prediction criterion. Informally, the selected parameter is $\widehat{\theta}$, where

$$\widehat{\theta} \approx \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} l(Y_i, m(\mathbf{x}, \theta)), \tag{4.2}$$

where l is some appropriate loss function (a semi-metric, that is a positive function such that $l(y, y) = 0$). Here, let us note that there is no guarantee that this $\arg\min$ is unique. Moreover, the approximation symbol in (4.2) is due to the fact that this minimum may not be exactly achieved, since relying on an algorithmic optimization.

Regarding the loss function the cross-entropy that we want to minimize is then computed from the duplicated dataset, defining

$$L_{n'}(p) = -\frac{1}{n'} \sum_{i=1}^{n'} \left\{ J_i' \log p + (1 - J_i') \log(1 - p) \right\}. \tag{4.3}$$

Remark. In the experimental part of the paper, we will also introduce an alternative loss functions, which can be seen as modifications of cross-entropy. The Weighted Cross Entropy (see [156]) is a

weighted version of (4.3). Introducing a weight α define

$$\bar{L}_{n'}(p) = -\frac{1}{n'}\sum_{i=1}^{n'}\{J_i'\alpha\log p + (1-\alpha)(1-J_i')\log(1-p)\}\,. \tag{4.4}$$

The parameter α can be used to increase the importance of good prediction for one of the two classes, or to reflect its under-representation in the sample.

The class of functions $\{l(\mathbf{x},\theta) : \theta \in \Theta\}$ is particularly rich, which offers an important flexibility to approximate many phenomena. On the other hand, increasing the number of neurons and/or layers make them prone to overfitting, due to the redundancy between the parameters. To overcome this difficulty, well-designed deep architectures are essential to improve the performance of such techniques, such as the Long Short Term Memory networks described in section 4.4.3.

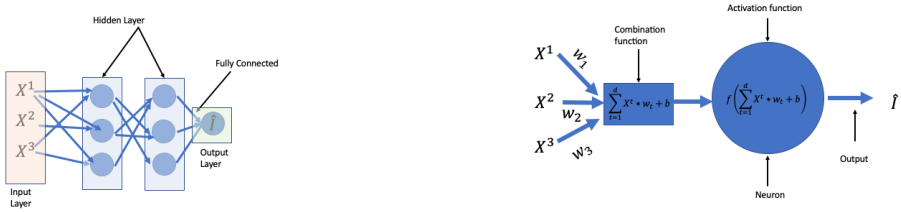


FIGURE 4.1 – Left-hand side : representation of a simple neural network. Each unit of the network (represented by a circle), is called a neuron (a neuron is represented on the right-hand side). $X^{(j)}$ represents the $j-$th component of the covariate vector $\mathbf{X}$.

4.4.2 Embedding layer

Text variables are difficult to handle due to the complexity of such unstructured data. The most simple way to encode text consists in using one-hot encoding, see for example [38]. One first determines a dictionary size V (after operating pre-treatments to avoid misspelled words or mistakes), which corresponds to the total number words (or sequences of words) in the whole corpus of texts. Each word is affected to a number between 1 and V. Each report is then encoded as a matrix. The $j-$th column corresponds to the $j-$th word in the report. If this world is the $k-$th word of the dictionary, all components of this column are 0, except for the $k-$th line where a 1 is put.

This representation is not optimal, in the sense that it does not define a proper distance between words. The idea behind embedding is to perform dimension reduction, and project the one-encoded matrices into a space which (ideally) takes into account the similarity between words. This projection is done via the application of a embedding matrix W to a one-encoded matrix $\mathbf{X}$.

This phase can be understood as a first layer of the global network, with a single neuron and an activation function which is identity. Rather than directly optimizing the coefficients, a preliminary initialization of this embedding matrix is essential. We rely on FastText algorithm described in [109]. This ensures that the embedding matrix has reasonable coefficient values before starting the optimization of the other parameters of the network, while allowing to update these coefficients through the global optimization process.

Let us observe that FastText also provides a predetermined embedding matrix, trained on a huge vocabulary. Nevertheless, the vocabulary of insurance reports is specific. Therefore it seems more adapted to train a new matrix on the corpus of reports to take this specificity into account.

4.4.3 Long Short Term Memory

Long Short Term Memory (LSTM) is a class of Recurrent Neural Networks introduced by [99]. They are called recurrent in the sense that they are constituted of successive blocks of neurons, the output of a block being one of the inputs of the following one. The network "reads" a report used as input respecting the order of the words : the first word (i.e. the first column of the embedded matrix) enters the first block of the network, the second word enters the second block, and so on.

The concept of memory is linked to the fact that the output of a block contains information on previous words which is kept, since sent to the next block. Nevertheless, keeping this information only through the output of a block could lead to a dilution of the influence of the $j-$th word in producing the output of the block $j+k$ if k is large. The particularity of LSTM networks is to introduce a conveniently designed "memory cell" c_t in addition to the output. This memory cell is designed to be able to keep relevant information, and to "forget" non essential : it is a vector which passes some so-called "gates" under which simple operations are performed to mimic these cognitive phenomena. This introduces more degrees of freedom to convey information.

A typical LSTM architecture is described in Figure 4.2.

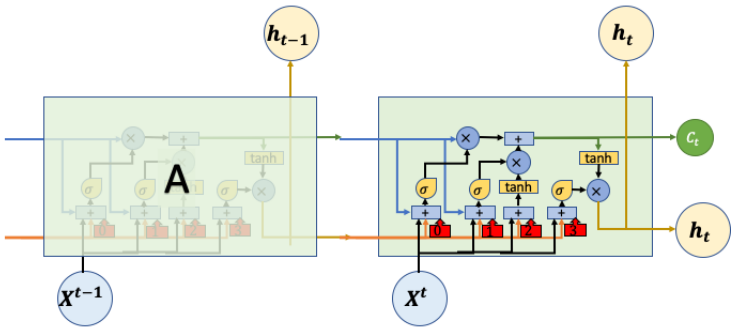


FIGURE 4.2 – Synthetic description of a block of a Long Short Term Memory network.

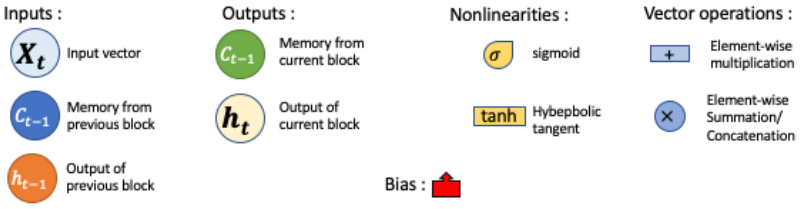


FIGURE 4.3 – LSTM and Bi-LSTM notations

To describe more in details Figure 4.2, the memory cell c_t, which is a vector, is first sent to a multiplicative "gate". This gate consists in the multiplication of each component of the cell by a different number between 0 and 1. This has to be linked with the idea of "keep information" Vs. "forget" : if one multiplies by a number close to 0, the information contained in the corresponding component of the cell is removed. This multiplicative vector, that we denote f_t in the following, is computed from the word x_t and the previous output h_{t-1}.

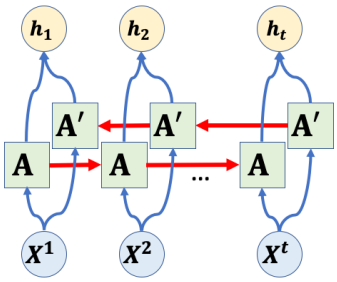


FIGURE 4.4 – Short description of a Bi-LSTM network. Each block A has the same structure as in Figure 4.2.

The modified memory cell then enters an additive gate used to incorporate new information, computed from x_t and h_{t-1}. This modified memory cell is the one that is sent to the next block. It is also used to compute the output of the current block, combining its contribution to x_t and h_{t-1}.

To summarize, the following set of operations are performed in each unit (here, σ denotes the classical sigmoid function, and $[x_t, h_{t-1}]$ the concatenation of the vectors x_t and h_t) :

- Compute $u_t = \sigma(\theta_0[x_t, h_{t-1}]+b_0)$, $v_t = \sigma(\theta_1[x_t, h_{t-1}]+b_1)$, and $w_t = \tanh(\theta_2[x_t, h_{t-1}]+b_2)$.
- The updated value of the memory cell is $c_t = u_t \times c_{t-1} + v_t \times w_t$.
- The new output is $h_t = \sigma(\theta_3[x_t, h_{t-1}] + b_3) \times \tanh(c_t)$.

The structure of LSTM makes them particularly well designed to handle text. Many applications of their use in text mining can be found in the literature, see for example [42]. For temporal data, we will use bidirectional LSTM (Bi-LSTM), which seem adapted to a time-evolving information. Their structure is described in Figure 4.4. Combining two independent LSTM together is a way to obtain both backward and forward information about the sequence at every time step. Doing so, a better understanding of the context is expected.

The particular structures of LSTM and Bi-LSTM that we use are described in the real data analysis section, see Figure 4.4. A multilayer perceptron is added to the end of the LSTM network to perform some synthesis of the outputs.

4.4.4 Extraction of features from the networks

As mentioned in section 4.3.2, the neural networks we use are used for features extraction from the set of covariates $(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{Z})$. There are, in the first step of our procedure, three networks that produce as outputs first predictions of the value of the variable I. These predictions are based, each time, only on a subset of the covariates, without considering interactions between them. Hence, less than these predictions by themselves, we are interested in the path the covariates goes through in the networks.

The final stage of this trajectory is contained in the last layer of the network, that is the set of neurons that are directly connected to the final output. The output of these neurons constitute the features we use in the next step.

4.5 Real data application

In this section, we show an illustration of the method on a real database. A short description of this database is given in Section 4.5.1. The analysis of the tail of the distribution of the loss variable is given in Section 4.5.2. We give details on the prediction of the severity of a claim in Section 4.5.3 where we predict the state I of a claim. Section 4.5.4 describes the architectures of the prediction models, while Section 4.5.5 shows the result on the final predicted amount.

4.5.1 Description of the database

The database we consider gathers 129,430 claims from a French portfolio from car insurance, separated between a train set (104,835 claims), a development set (12,952 claims) and a test set (11,643 claims). A "severe" claim is defined as a claim with loss upper than 25000€. In Table 4.1, we see that a "severe" event is rare. Moreover the time of development is 22 month (in average).

	Proportion of severe	Average duration	Censored claims
Train	2.67%	10.84	13.16%
Development	2.76%	10.81	12.97%
Test	2.91%	10.91	13.31%

TABLE 4.1 – General statistics on the database.

For each claims, 709 variable are present. Among these variables, 437 do not evolve with time, and are related to the contract itself (type of guarantee, characteristics of the policyholder...) or to the type of the claim (material or corporal damages for instance). Additionally, 172 variables reflect the evolution of the claim and 100 are from the embedding matrix.

For example,
- rate of disability of the victim(s)
- induced actual revenue losses
- induced future revenue losses
- health costs
- third party assistance
- permanent esthetic prejudice...

The variables related to the health expertise are filled after a medical expertise. Several medical expertises be sent through the lifetime of the claim, reevaluating the values of the corresponding informations. For example, Figure 4.5 shows the evolution of the rate of disability (which, in France, is called DFP for "Déficit Fonctionnel Permanent") for four victims. Two are related to "severe" claims, while the two others are not. In some cases, the evaluation of this rate stabilises. But in other ones, many fluctuations occur before a stabilisation (only 1 year before closure in one of the examples).

Additionally, we have external data describing the environment (like, for example, the sinistrality at a local level, extracted for example from open databases like URL). These variables are not directly linked to the claim but bring some potential information about the context or the dangerousness of the location where the claim occurred.

About the reports, they have been established at the opening of the claim, and do not take into account counter-expertises. Let us give two examples of phrases for each category (standard claim and severe claim) :

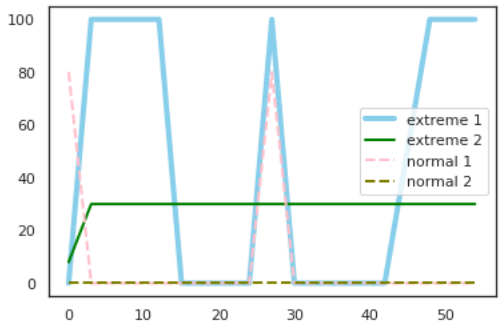


FIGURE 4.5 – Evolution of the disability rate (DFP) of 4 individuals through time. The label "extreme" indicates a severe claim, "normal" indicates a standard claim.

— standard claim :

1. "pile-up on the road, 58 vehicles involved."
2. "insured gets hit at the rear, shock to the rear bumper."

— extreme claim :

3. "insured hits the solid ground of a roundabout and dies during the transfer to emergency."
4. "third party does not stop at a stop sign and hits the insured in the front right, air bag triggered."

As one can see from these example, one of the specificity is that reports are often written in a short way, with a syntax which can sometimes be very brief. The vocabulary is also very specific to car insurance. On the other hand, in the third example, one can see that the conclusion that the claim is severe is obvious, since someone died. But example 4 is less obvious. A shock in the front is reported, but without explicit reference to the state of the victims, identification of such a claim as severe is expected to be much harder.

4.5.2 Tail of the distribution of the loss

In this section, we take a closer look to the tail of the distribution of the loss variable L. This analysis is made on the uncensored observations only. A Hill plot is shown in Figure 4.6. From this graph, we see that the Hill plot is close to linearity for a threshold $l_0 = 25000$ The idea is then to separate the distribution into two parts : a (truncated) GLM for "standard" claims ($I = 1$ when $L \leq l_0$), and a Generalized distribution for "extreme" ones ($I = 2$ when $L > l_0$).

Introducing the survival function of a Generalized Pareto distribution, that is

$$S_{\gamma,\sigma}(y) = \left(1 + \gamma \frac{y}{\sigma}\right)^{-1/\gamma}, \ y > 0., \tag{4.5}$$

we estimate the parameters (γ, σ) using (pseudo-)maximum likelihood estimation, where the observations have been weighted to correct the bias caused by the censoring (using the algorithm 7). The values of the estimated parameters are $\widehat{\gamma} = 7.831843e - 01$ (95% confidence interval : $[0.7830573; 0.7833113]$), $\widehat{\sigma} = 3.250866e + 04$ (95% confidence interval $[32506, 32510]$).

FIGURE 4.6 – Hill plot for the variable L. On the top, value of the threshold; on the bottom, corresponding rank of the observation in the sample.

4.5.3 Prediction models

The summary of the whole procedure of section 4.2 is shown in Figure 4.7.

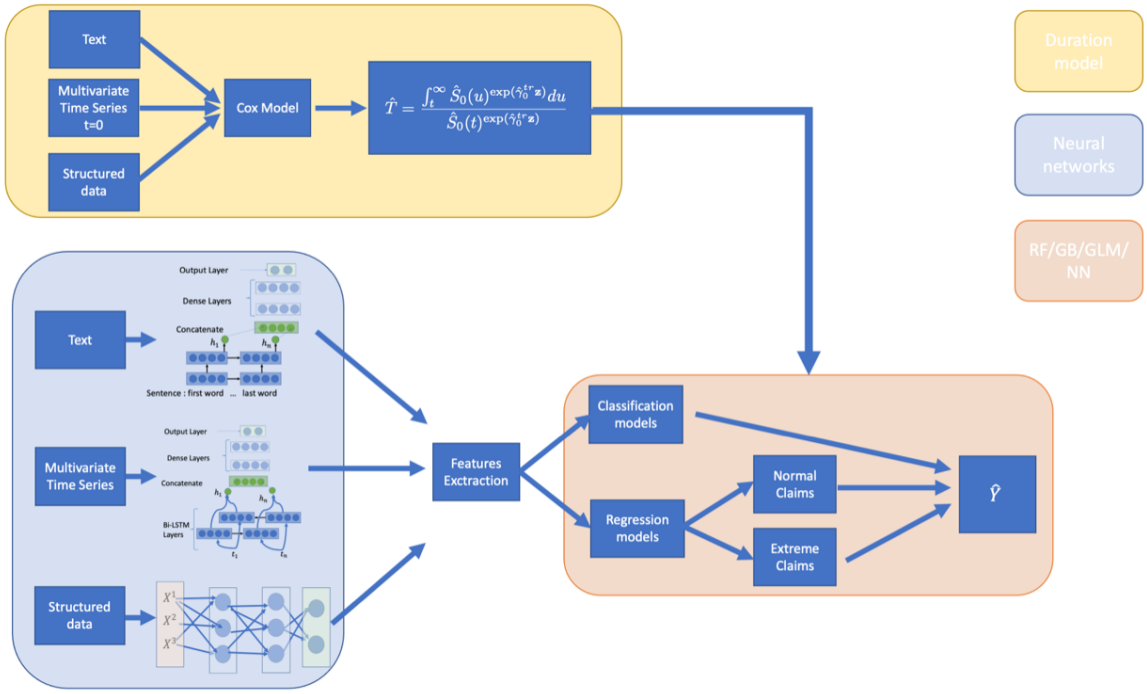


FIGURE 4.7 – Summary of the whole procedure. The yellow box corresponds to the prediction model on T. The blue box represents the different neural network structures used to perform feature extraction. The orange box gathers these two inputs and combines a prediction of a severity (using different machine learning techniques) with a model to predict the cost.

The hyperparameters for the different procedure that are used are summarized below. These hyperparameters have been selected through a grid search. More details on this grid search are available on GITHUB. `https://github.com/isaaccs/Prediction-of-the-outcome-of-insurance-claims-with-deep-neural-networks/blob/main/hyperparameters.md`

4.5.4 Classification of claims

In this section, we investigate the ability of the procedure to detect if a claim belongs to category $I = 1$ or $I = 2$. The proportion of labels $I = 2$ is low in the sample (2.7%) compared to the proportion of 1. For this reason, the criterion that we will use to determine the quality of our predictions is the F_1-score.

The F_1-score (see for example [4]) is defined as We then define Recall and Precision as

$$F_1 = \frac{2 \times Recall \times Precision}{Recall + Precision},$$

where

$$\begin{aligned} \text{Recall} &= \frac{TP}{TP + FN}, \\ \text{Precision} &= \frac{TP}{TP + FP}, \end{aligned}$$

and where TP is the number of true positive, FN of false negative, FP of false positive.

The results regarding the F_1-score for the different models is reported in Table 4.2 below.

Year	NN	RF	GB	GLM
0	0,69	0,72	0,70	0,71
1	0,73	0,74	0,73	0,77
2	0,76	0,55	0,77	0,75
3	0,79	0,79	0,78	0,79
4	0,80	0,79	0,84	0,85
5	0,82	0,75	0,85	0,83
6	0,85	0,80	0,83	0,87
7	0,84	0,86	0,81	0,76
8	0,90	0,90	0,68	0,81

TABLE 4.2 – F_1-score of the different prediction models for different maturities of claims.

The information on the prediction can also be retrieved from the normalized confusion matrices, see Figure 4.8.

Year	NN	RF	GB	GLM
0	0,40	0,45	0,41	0,43
1	0,49	0,49	0,49	0,56
2	0,56	0,14	0,58	0,53
3	0,65	0,62	0,62	0,63
4	0,67	0,65	0,73	0,75
5	0,72	0,61	0,66	0,73
6	0,76	0,67	0,75	0,79
7	0,75	0,81	0,75	0,61
8	0,83	0,83	0,45	0,71

TABLE 4.3 – F_1-score of extrems claims of the different prediction models for different maturities of claims.

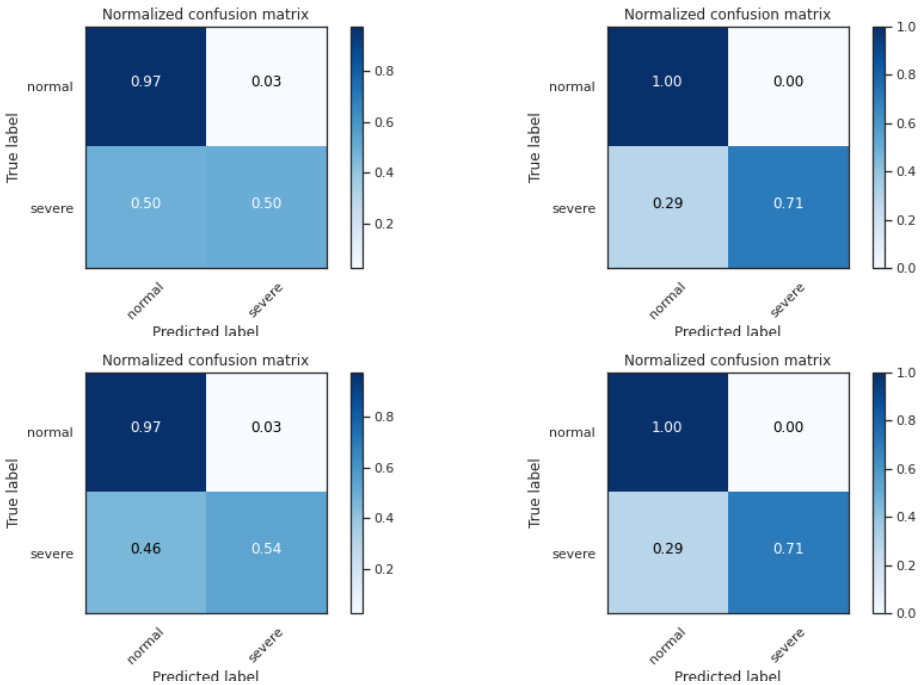


FIGURE 4.8 – On the top left corner, normalized confusion matrice for our neural network at the opening of the claim, on the top right, at the end of the claim. Below, the same timestamp but for our GLM

One can see that, especially for claims which have been recently opened, many severe claims are not predicted as such. This is expected, since the definition we took for a "severe" claims, corresponds to a high upper quantile of the distribution of the losses, hence the prediction problem is difficult since few such claims are present in the database used to calibrate the model.

We can see that the GLM and Neural Networks procedures provide better results. The detection of severe claims increases when the claim is open from a long time. On the other hand, the random forests and gradient boosting methods are significantly less performant to perform this detection.

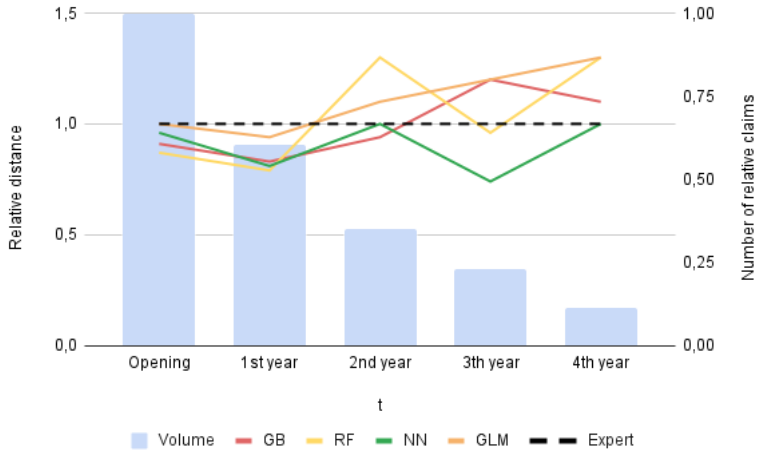


FIGURE 4.9 – Relative comparison of claims models against the expert's predictions in the first 4 years.

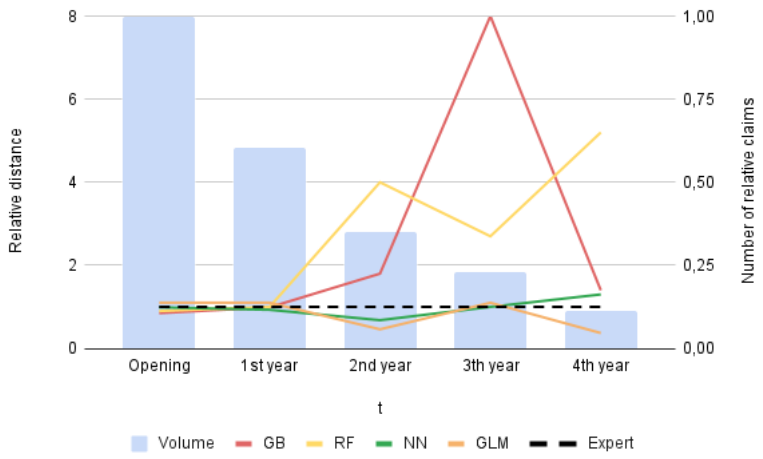


FIGURE 4.10 – Relative comparison of extreme claims models against the expert's predictions in the first 4 years.

4.5.5 Claim amounts and reserving

We now combine the prediction results with the models on the cost. For the GLM use case, if a standard claim is predicted ($\widehat{I}_i = 0$), a Gamma model is used to predict the final amount. Otherwise, the average value of the fitted Generalized Pareto distribution is used.

The results are shown in Table 4.4

	compare to	**mean T**			**true T**			**with out T**		
year	Model	global	severe	normal	global	severe	normal	global	severe	normal
0	GB	1.0	1.0	1.01	0.94	0.86	1.6	1.02	1.03	0.97
	GLM	1.0	1.0	1.04	1.0	1.0	1.02	1.0	0.98	1.63
	NN	1.0	1.0	1.01	1.0	1.0	0.98	1.02	0.99	1.53
	RF	1.01	1.0	1.17	1.0	1.0	1.0	0.88	0.86	1.07
1	GB	1.0	1.0	1.0	1.0	1.0	1.0	1.15	0.99	1.18
	GLM	1.0	1.0	1.0	1.0	1.0	1.0	1.01	1.1	1.0
	NN	1.0	1.0	1.0	0.94	1.0	0.92	0.92	1.08	0.9
	RF	1.0	1.0	1.0	1.0	1.0	1.0	0.97	1.29	0.94
2	GB	1.0	1.0	0.97	1.0	1.0	0.97	0.93	3.86	1.09
	GLM	1.0	1.0	1.01	1.0	0.98	1.01	0.99	1.07	1.07
	NN	1.0	1.0	1.01	1.0	1.0	1.0	0.8	0.1	0.22
	RF	0.71	0.33	0.19	1.0	1.0	1.0	0.51	0.21	0.14
3	GB	1.13	2.27	1.51	1.0	1.0	1.0	1.48	1.19	2.08
	GLM	1.0	0.88	1.02	1.0	0.99	0.98	0.96	1.2	0.91
	NN	1.0	0.99	1.0	1.0	1.01	1.0	1.24	1.48	0.9
	RF	1.01	1.0	1.08	1.0	1.0	1.0	0.99	0.87	0.79
4	GB	0.99	1.13	1.48	1.0	1.16	1.02	0.97	1.54	1.21
	GLM	1.0	1.0	1.0	1.0	0.96	1.0	0.99	0.98	0.8
	NN	1.0	1.01	1.17	1.0	1.0	1.0	0.82	2.72	1.32
	RF	1.04	1.02	1.27	0.99	1.0	1.0	0.97	0.27	0.55
5	GB	1.0	1.0	0.94	1.0	0.87	0.68	0.96	0.54	0.87
	GLM	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.94
	NN	1.0	1.0	0.99	1.0	1.0	0.98	0.99	1.18	1.0
	RF	1.0	1.01	1.02	1.0	1.0	1.02	1.01	1.55	2.15
6	GB	1.0	1.0	1.0	1.0	1.0	1.0	1.17	21.05	5.14
	GLM	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.6	0.98
	NN	1.0	1.0	1.0	1.0	1.0	1.0	0.82	0.3	0.67
	RF	1.0	1.0	1.47	1.0	1.0	1.0	0.96	0.04	0.24
7	GB	1.0	0.79	0.96	1.0	1.02	1.0	0.96	1.14	3.25
	GLM	1.0	1.0	1.0	1.01	1.0	0.77	1.0	0.15	0.82
	NN	1.0	1.0	1.0	1.0	1.0	1.0	0.97	1.01	1.31
	RF	1.0	1.0	1.01	1.0	1.0	1.06	1.08	0.05	0.22
8	GB	1.0	1.02	0.99	1.0	1.03	1.0	0.81	0.02	0.15
	GLM	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.83
	NN	1.0	1.04	1.0	1.0	1.01	1.0	0.8	1.05	0.93
	RF	1.0	1.49	0.94	1.0	1.48	1.0	1.01	0.15	0.47

TABLE 4.4 – MSE comparison of our models with $\widehat{T}$ against the mean T, the true T and without T

We can see that our modelisation of T helps our model to have a good performance. Close to have the real information.

4.6 Conclusion

In this work, we show how artificial intelligence techniques can be used to improve the analysis of claims based on complex data. The procedures gathers different treatment of data depending on the characteristics of the input variable (evolving through time or not, text-based data). A prediction of the time of settlement of the claims is also used, since this variable is known to have a significant importance on the final amount. In the example we study, the tail of the distribution of the losses is important, as often when there is a matter of corporal damage. A particular treatment of "extreme" claims is performed. Let us point that, although the methodology improves the final vision of the amount, the problem is still complex, and the fact that the event we are focusing on are rare (a few percents of the claims) introduces lots of variability in the prediction. However, the deep learning methods that we use significantly improve the rate of detection of such claims, and allow to have a clearer view on the final reserve.

Chapitre 5

Conclusion et perspectives

L'amélioration du provisionnement permet de diminuer les incertitudes économiques liées aux engagements pris. La réduction de cette incertitude ouvre des perspectives considérables aussi bien en terme d'augmentation de la rentabilité, que de compétitivité commerciale : en réduisant la marge d'erreur autour de l'estimation de la charge - laquelle peut se révéler considérable pour certaines garanties telles que la responsabilité civile - on est en mesure de dégager un certain montant de capital.

Le but de cette thèse était de montrer comment la richesse des données disponibles sur des sinistres d'assurance peut être mise à profit pour améliorer significativement la prédiction du montant final d'un sinistre (ou son issue, lorsque l'on s'intéresse notamment à la classification d'un sinistre suivant son niveau de gravité). Pour traiter un tel volume de données, dont certaines sont peu usuelles en assurance comme les données textuelles, nous avons eu recours à des techniques d'apprentissage de type Machine Learning (notamment des réseaux de neurones profonds tels que les Convolutional Neural Networks ou des réseaux Long Short Term Memory) à la fois en tant que prédicteurs, mais aussi en tant qu'extracteurs d'information.

L'étude des sinistres corporels nécessite un traitement spécifique du fait de leurs caractéristiques et de l'extrême volatilité de leur coût. Notre problématique concernait principalement les sinistres graves, ces sinistres peu nombreux qui pèsent extrêmement lourd sur les comptes de l'assureur. L'utilisation d'outils issus de la Théorie des Valeurs Extrêmes (voir Chapitre 2) nous a permis d'analyser la queue de distribution du montant des sinistres, mais aussi de déterminer un seuil de gravité. Cette séparation des données vise à améliorer la modélisation d'observations extrêmement volatiles aux caractéristiques différentes de celles du cœur de la distribution. Une autre spécificité de notre approche a été de prendre en compte l'écoulement temporel des sinistres, particulièrement important lorsque l'on s'intéresse à une branche d'assurance à développement long comme la responsabilité civile. Au cours de cette thèse, nous avons été amenés à plusieurs reprises à utiliser les poids IPCW, Inverse-Probability-of-Censoring Weighting, afin de traiter le phénomène de censure qui rend l'information disponible incomplète.

Une fois le seuil de gravité estimé et la censure corrigée, il a été nécessaire d'explorer les données mises à disposition par l'assureur et de définir un périmètre d'étude. En faisant l'inventaire des données disponibles, les rapports de sinistres nous ont paru constituer un matériau précieux. Ces rapports sont établis à l'ouverture du sinistre par les gestionnaires et demandent un prétraitement afin d'être utilisable par nos modèles. Ces données ont attiré notre attention car elles sont, pendant quelques mois, la seule information que possède l'assureur et n'ont jamais été exploitées par celui-ci. Par ailleurs, leur traitement automatique nous paraissait intéressant, car leur exploitation pratique

par les gestionnaires sinistres est parfois délicate et peut donner lieu à de grandes variabilités de traitement.

Suite au recueil des données, nous avons travaillé sur deux méthodologies afin d'améliorer le provisionnement. La première, développée au chapitre 3, vise à classifier automatiquement ces rapports de sinistres. L'utilisation de méthodes de type réseaux de neurones combinées à différents algorithmes de rééquilibrage nous a permis de prédire si un sinistre est grave ou non. Notre étude a montré que l'utilisation du texte à l'ouverture du sinistre offre à l'assureur la possibilité d'améliorer significativement la détection de sinistres extrêmes. Cette amélioration de la prédiction dès l'ouverture évite à l'assureur d'être pris de court durant la vie du sinistre. La deuxième méthode quant à elle, présentée au chapitre 4, prend en compte l'aspect temporel du sinistre ainsi que ses variations et vise à estimer le coût final d'un sinistre ouvert. Cette fois-ci, nous avons pris en compte la diversité des covariables (structurées et non structurées), ce qui nous a amené à combiner à la fois différents types de réseaux de neurones afin de réduire la dimension de notre problème, mais également des méthodes de prédiction d'analyse de survie afin de prédire le moment du règlement du sinistre. Une fois les caractéristiques de sinistres extraites, on peut les connecter à un réseau de neurones final ou à des modèles plus interprétables, de type GLM, si le besoin d'interprétation est plus important que la qualité de la prédiction. En plus d'améliorer la prédiction de la charge finale, il est intéressant de constater que cette méthode valide notre intuition développée au chapitre précédent : le texte qui a un fort pouvoir explicatif à l'ouverture du sinistre se voit avec un pouvoir prédictif de plus en plus faible au bénéfice des variables de sinistralité qui, quant à elles, évoluent au fil du temps.

Cette thèse offre de nombreuses perspectives en termes de recherche que l'on pourrait résumer en deux parties : d'une part les modèles adaptés à ces questions de provisionnement, de l'autre les données.

En effet, les modèles de Deep Learning cherchant à modéliser du texte évoluent continûment. La comparaison de différentes méthodes d'Embedding à la place de la méthode Fasttext que nous avons utilisée pour traiter et transformer les rapports de sinistre pourrait s'avérer intéressante. Les modèles de type Transformers et utilisant le mécanisme d'Attention, sont en effet plus abouties en termes de prise en charge du contexte. De plus, la couche d'Attention apporte une part d'interprétabilité au modèle et permet donc à l'assureur d'avoir une meilleure lecture et une meilleure compréhension des résultats. La méthode mise en place au Chapitre 4 est certes performante, mais présente l'inconvénient d'entraîner un nombre de modèles proportionnel au nombre de pas de temps. Afin d'alléger l'étape d'entraînement, il pourrait être pertinent de tester des modèles LSTM type $N \times M$ plutôt que M modèles de type $N \times 1$ comme nous l'avons fait. Un autre axe d'amélioration, qui permettrait également d'alléger l'architecture globale du modèle, serait d'utiliser un MDN (Mixture Density Network) qui prédirait directement le coût final et non pas trois modèles comme c'est le cas actuellement. Mis à part l'architecture elle-même, une modélisation d'autres variables de durée, telles que le temps de consolidation, pourrait aider à améliorer la modélisation de la durée du sinistre et donc également la prédiction de la charge finale. Nous retenons donc que l'utilisation d'une génération plus récente de réseaux de neurones et de méthodes d'Embedding plus élaborées ainsi que la prise en charge de l'aspect dynamique des séries temporelles représentent un axe d'amélioration majeur à nos travaux de recherche.

Sur le plan de la donnée, cette thèse a montré que les données non structurées aidaient à la modélisation du coût et de la gravité du sinistre. Il peut être intéressant d'utiliser d'autres types de données, telles que les images par exemple, afin d'aider l'assureur dès l'ouverture du sinistre comme peut le faire le texte. Concernant les données externes, nous les avons traitées comme des variables statiques, c'est-à-dire figées dans le temps. Or ces données sont mises à jour de façon régulière (dont

certaines chaque année). Il pourrait être intéressant d'utiliser ces données sous forme de séries temporelles afin d'avoir une meilleure compréhension de l'évolution du risque géographique.

Tous les axes d'amélioration énumérés dans cette conclusion peuvent être à l'origine de nombreux travaux auxquels nous aimerions contribuer.

Annexe A

Chapitre 1

A.1 Descente de Gradient : Méthode de Newton-Raphson

La descente de gradient est une technique itérative qui permet d'approcher la solution d'un problème d'optimisation. En apprentissage supervisé, la construction du modèle revient souvent à déterminer les paramètres (du modèle) qui permettent d'optimiser (max ou min) une fonction de coût. On utilise une descente de gradient dans le Gradient Boosting, mais aussi dans les GLM. Il existe plusieurs types de méthode, je ne présenterai que celle de Newton-Raphson qui est l'une des plus utilisées et dont la technique est très proche des autres modèles de descente de gradient.

A.2 Méthode de Newton-Raphson

Une procédure numérique pour maximiser la fonction de log-vraisemblance $l(\theta)$ est la méthode de Newton-Raphson, qui permet de déterminer un point où le gradient $\Delta l(\theta)$ s'annule. Ce n'est pas équivalent à déterminer le point du maximum global d'une fonction, car ce point peut être un maximum local seulement, un minimum ou un point d'inflexion. Mais, dans de nombreux cas, cette méthode donne des résultats satisfaisants pour détecter le point maximum d'une fonction. C'est une procédure itérative (On parle aussi de descente de gradient stochastique) pour trouver des points critiques d'une fonction réelle $f : \chi \to \mathbb{R}$ où $\chi \subset \mathbb{R}^d$. f est supposée deux fois dérivable. Cette méthode repose sur l'approximation linéaire du gradient $\Delta f(x)$ par

$$\Delta f(x) = \Delta f(\xi) + H(\xi)(x - \xi) + r(x, \xi)$$

où $\xi \in \chi$ et $H(\xi) = \Delta^2 f(\xi)$ la matrice hessienne de f en ξ et $r(x, \xi)$ un terme de reste. Si ξ est proche de x le reste est négligeable et c'est sur quoi repose la méthode de Newton-Raphson car on ne résout pas $\Delta f(x)$ mais $\Delta f(\xi) + H(\xi)(x - \xi) = 0$ par rapport à x. La méthode étant itérative, on l'initialise en posant $\xi = x^{(0)}$ puis on résout l'équation, la solution est notée $x^{(1)}$ et on pose $\xi = x^{(1)}$ et cela jusqu'à convergence de l'algorithme. D'où

$$x^{(t)} = x^{(t-1)} - [H(x^{(t-1)})]^{-1} \Delta f(x^{(t-1))}$$

où $x^{(t-1))}$ est le résultat de l'itération précédente. De plus, on voit que pour l'algorithme soit bien défini, il faut que $[H(x^{(t)})]^{-1}$ existe pour tout t.

A.2.1 Interprétation Géométrique

En remarquant que le terme à gauche de $\Delta f(\xi) + H(\xi)(x - \xi) = 0$ est la tangente de $\Delta f(x)$ au point $x = \xi$. Au lieu de chercher la solution de $\Delta f(x) = 0$ on cherche le zéro de la tangente. Et comme la tangente est linéaire, il est plus facile de trouver son zéro que le celui de $\Delta f(x)$.

A.2.2 Critère d'arrêt

Il existe plusieurs critères d'arrêt de l'algorithme de Newton-Raphson, mais les deux plus utilisés sont les suivants. Soit $\epsilon > 0$ un seuil fixé, on arrête l'algorithme dès que :

- $|x^{(t))} - x^{(t-1))}| < \epsilon$
- $|\Delta f(x^{(t))})| < \epsilon$

Attention, peu importe le critère d'arrêt, il se peut que la condition soit vérifiée dans des points ne correspondant pas à des zéros de $\Delta f(x)$. Il est en général difficile de garantir la convergence de la suite $x^{(t))}$. En revanche, si elle converge, elle converge assez vite et c'est ce qui rend la méthode de Newton-Raphson intéressante.

A.3 Évaluation d'un modèle de Régression

Il s'agit d'évaluer la qualité d'ajustement du modèle sur la base des différences entre observations et estimations.

A.3.1 GLM

Plusieurs critères existent pour évaluer la qualité d'un GLM.

A.3.1.1 Déviance

Le modèle estimé est comparé avec le modèle dit saturé [1]. Cette comparaison est basée sur l'expression de la déviance noté D des log-vraisemblances $\mathscr{L}$ et $\mathscr{L}_{sat}$:

$$D = -2(\mathscr{L} - \mathscr{L}_{sat})$$

qui est le logarithme du carré du rapport des vraisemblances. Ce rapport « généralise »l'usage des sommes de carrés propres au cas gaussien et donc à l'estimation par moindres carrés.

Un test de rejet du modèle peut être construit en montrant asymptotiquement D suit une loi du χ^2 à $n - p$ paramètres. Mais cette approximation n'est pas forcément vraie dans le cas où les données sont groupées et dans ce cas le nombre de paramètres tend vers l'infini.

Dans notre étude, la déviance a été utilisée afin de juger la qualité des différents modèles de coût moyen que nous avons pu créer.

1. C'est le modèle possédant autant de paramètres que d'observations et estimant donc exactement les données.

A.3.1.2 Test de Pearson

Un test de χ^2 est également utilisé pour comparer les valeurs observées y_i à leur prévision par le modèle. La statistique du test est définie par

$$\mathbb{X}^2 = \sum_{i=1}^{l} \frac{(y_i - \widehat{\mu}_i)^2}{\widehat{\mathbb{V}}(\widehat{\mu}_i)}$$

et on montre qu'elle admet asymptotiquement la même loi que la déviance.

En général, ces deux méthodes nous donnent à peu de chose près les mêmes résultats. Si ce n'est pas le cas, cela indique une mauvaise approximation de la loi asymptotique. Sachant que l'espérance d'une loi du χ^2 est son nombre de degrés de liberté. Comme les tests construits sont souvent approximatifs, on regarde souvent le rapport $\frac{Stat\ de\ test}{Degre\ de\ liberte}$ pour juger si le modèle est satisfaisant ou non. Pour le savoir, le rapport doit être plus petit que 1.

Dans notre étude, le test de Pearson a été utilisé afin de juger la qualité des différents modèles de fréquence que nous avons pu créer.

A.3.1.3 AIC

Le critère d'information d'Akaike (AIC) est une mesure de la qualité d'un modèle statistique proposée par Hirotugu Akaike.
Lorsque l'on estime un modèle statistique, il est possible d'augmenter la vraisemblance du modèle en ajoutant un paramètre. Le critère d'information d'Akaike, permet de pénaliser les modèles en fonction du nombre de paramètres afin de satisfaire le critère de parcimonie. On choisit alors le modèle avec le critère d'information d'Akaike le plus faible.
Il se calcule de la façon suivante :

$$AIC = -2\log(L) + 2(k+1)$$

où k est le nombre de paramètres à estimer du modèle et L est le maximum de la fonction de vraisemblance du modèle.

A.3.2 Métrique utilisé pour les méthodes de Machine Learning

Il existe plusieurs méthodes pour évaluer la qualité d'un modèle de régression. Elles se basent toutes sur :

- y_i la valeur observée de la variable à prédire.
- $\widehat{y_i}$ la valeur prédite par le modèle pour un y_i.
- $\bar{y}$ qui est la prévision naïve de référence.
- $y_i - \widehat{y_i}$ l'erreur de prédiction du modèle.
- $y_i - \bar{y}$ l'erreur de prédiction naïve.

Tout cela permet de définir plusieurs indicateurs de performance du modèle, mais celui qu'on a utilisé est le coefficient de détermination noté R^2 est défini par

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(y_i - \widehat{y_i})^2}{\sum_{i=1}^{m}(y_i - \bar{y})^2}$$

Le R^2 varie entre 0 et 1, s'il vaut 35%, cela veut dire que 35% des variations de la variable Y sont expliquées par le modèle. Mais attention, un R^2 trop proche de 1 représente un risque de sur-apprentissage. Un autre problème de cet indicateur est qu'il a tendance à grandir quand la dimension augmente. Or, comme on le sait, un modèle avec beaucoup de variables explicatives est peu robuste. C'est pourquoi dans le cas d'une régression multiple, on a tendance à utiliser le $R^2_{ajusted}$ qui corrige le R^2 par rapport aux nombres de variables explicatives.

$$R^2_{ajusted} = 1 - \frac{(1 - R^2)(m - 1)}{(m - k - 1)}$$

où m est le nombre d'observations et k le nombre de variables explicatives dans le modèle.

Tandis que le R^2 permet de comparer deux modèles de même dimension, le $R^2_{ajusted}$ permet de comparer des modèles ayant un nombre de variables explicatifs différents.

A.3.3 MSE

En statistiques, l'erreur quadratique moyenne (ou plus souvent l'erreur quadratique, moyenne étant sous-entendue), appelée aussi risque quadratique, pour un paramètre θ de dimension 1, que nous noterons MSE (pour Mean Squared Error), est définie par : $\mathbb{E}((\widehat{\theta} - \theta)^2)$

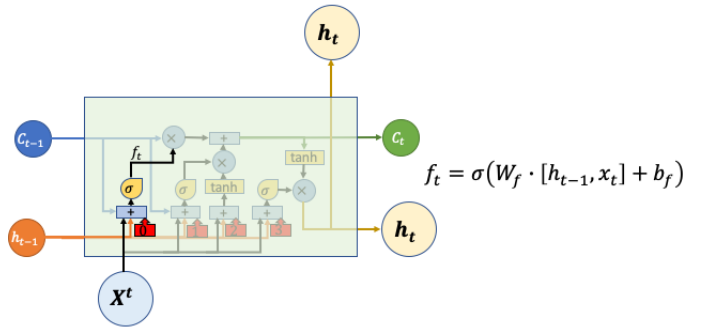
Cet estimateur mesure à quel point les prédictions sont proches des données observées. C'est la moyenne arithmétique des carrés des écarts entre les prévisions et les observations.

C'est la valeur à minimiser dans le cadre d'une régression simple ou multiple. La méthode est fondée sur la nullité de la moyenne des résidus. Mais la moyenne de leurs carrés n'est généralement pas nulle. Cette moyenne n'est autre que la VARIANCE RÉSIDUELLE que l'on cherche à minimiser (Cf. le théorème de König).

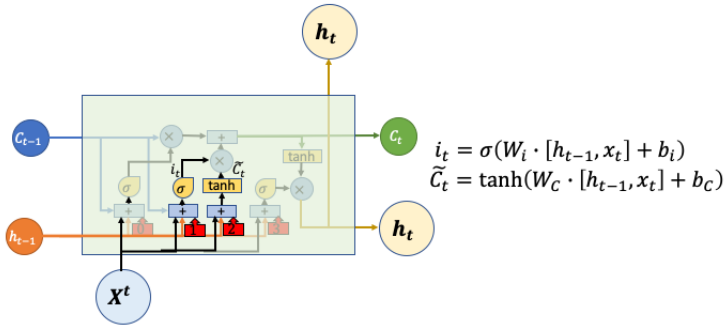
Sa formule est la suivante :

$$MSE = \frac{1}{m} \sum_{i=1}^{m} (y_i - \widehat{y_i})^2$$

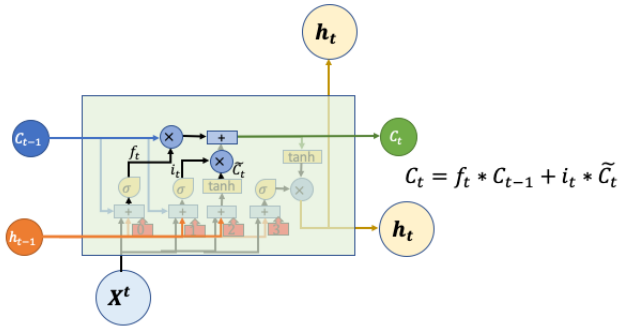
A.4 Les étapes du LSTM



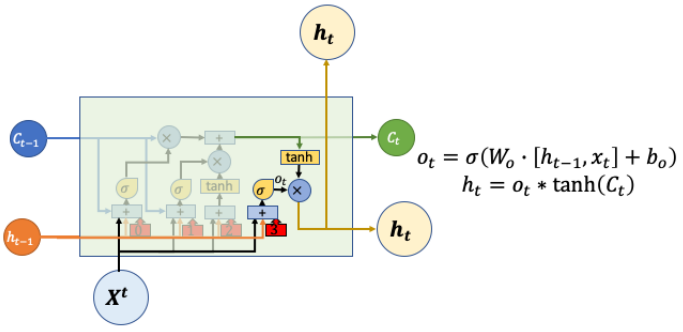
(a) Étape 1



(b) Étape 2



(c) Étape 3



(d) Étape 4

FIGURE A.1 – Les différentes étapes du LSTM

Annexe B

Annexe Chapitre 2

B.1 Graphiques complémentaires pour la TVE

B.1.1 2002-2006

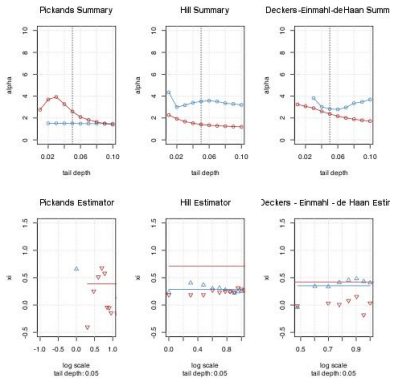


FIGURE B.1 – Les estimateurs

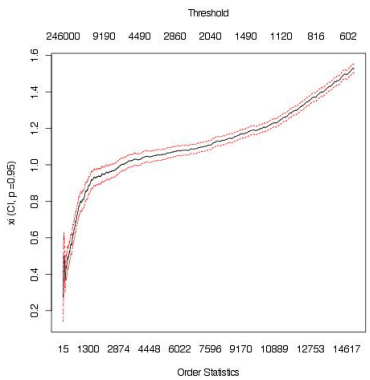


FIGURE B.2 – Graphique de Hill

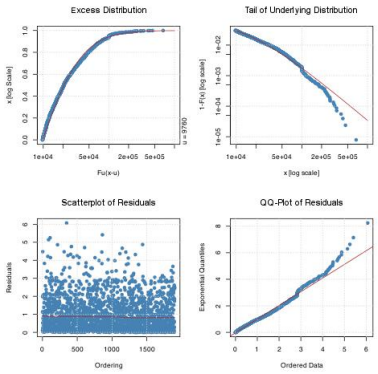


FIGURE B.3 – GPD

B.1.2 2007-2011

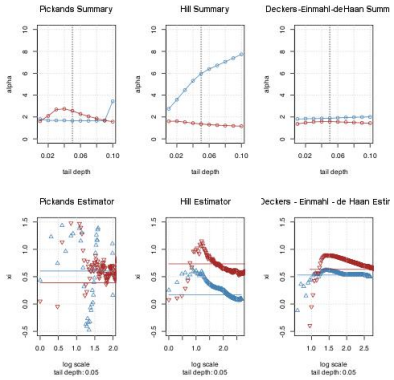


FIGURE B.4 – Les estimateurs

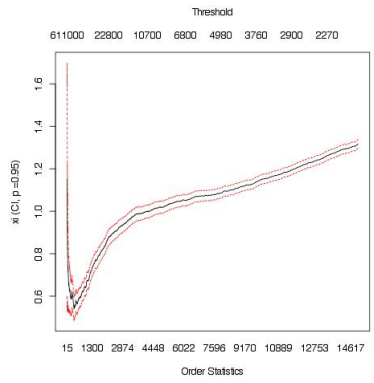


FIGURE B.5 – Graphique de Hill

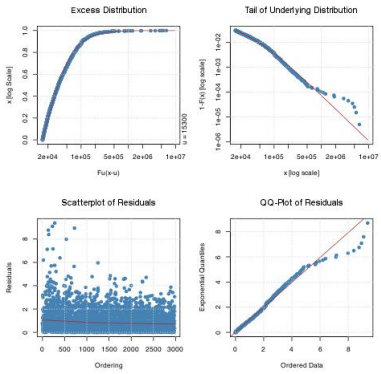


FIGURE B.6 – GPD

B.1.3 2012-2016

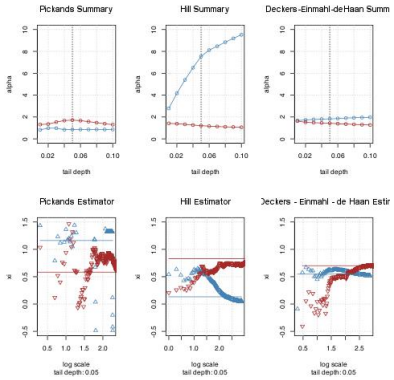


FIGURE B.7 – Les estimateurs

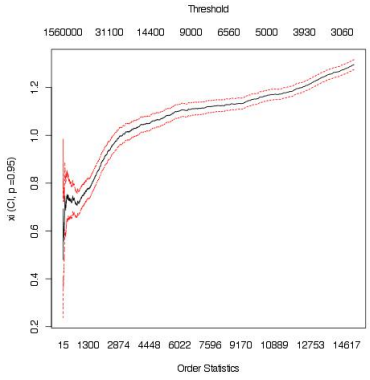


FIGURE B.8 – Graphique de Hill

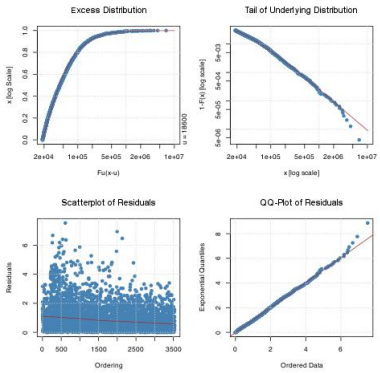


FIGURE B.9 – GPD

Annexe C

Chapitre 3

C.0.1 Typical choices of activation functions for neural networks

A list of typical activation functions is provided in Table C.1.

-1cm

Name	Function
Sigmoid function	$\sigma : \mathbb{R} \to [0,1]$, $x \mapsto \frac{1}{1+e^{-x}}$
Hyperbolic tangent	$tanh : \mathbb{R} \to [-1,1]$, $x \mapsto 2\sigma(2x)+1$
ReLU	$f : \mathbb{R} \to \mathbb{R}^+$, $x \mapsto max(0,x)$
Leaky ReLU	$f : \mathbb{R} \to \mathbb{R}^+$, $x \mapsto \alpha x \mathbb{1}_{x\leq 0} + x\mathbb{1}_{x>0}$ where $\alpha > 0$.
Swish function[169]	$\sigma : \mathbb{R} \to [0,1]$, $x \mapsto \frac{x}{1+e^{-x}}$

Table C.1 – Typical choices of activation functions.

C.0.2 Additional type of layers in a CNN

C.0.2.0.1 Zero-padding.

Zero-padding is the simplest way to avoid diminishing too much the dimension of the input when going deeper into the network. As we can see above in Figure 3.3, a convolution step will produce an output C of smaller dimension than A. Moreover, when dealing with our text data, it is useful to control the size of our inputs. Let us recall that we want to automatically analyse reports to predict the severity of a claim. These reports do not all have the same size in terms of words. Zero-padding creates a larger matrix by adding zeros on the boundaries, as shown in the example of Figure C.1. Similar issues are present when dealing with images that may not all have the same number of pixels.

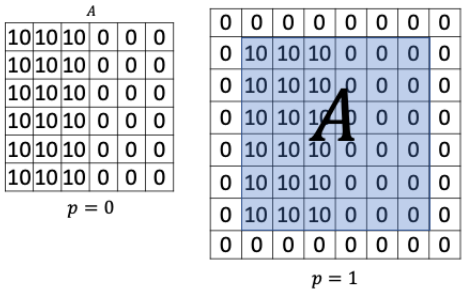


FIGURE C.1 – Zero-padding

C.0.2.0.2 Pooling Step.

Pooling steps work similarly as convolution layers, but applying locally a function that may not be linear. An example is shown in Figure C.2.

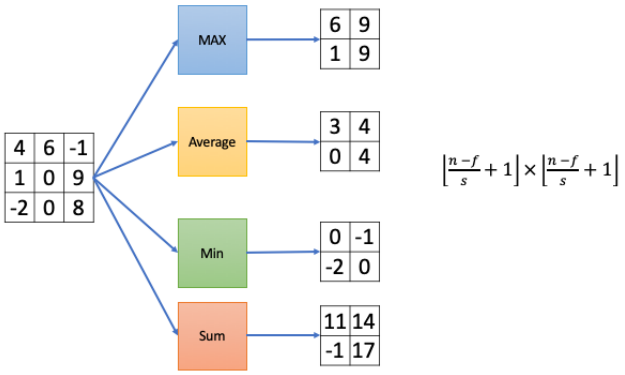


FIGURE C.2 – Example of a Pooling Layer : each function is applied to a moving square of input coefficients of size 2x2.

Let us note that, in the example of Figure C.2, the functions 'Average' and 'Sum' may be seen as linear filters, as in Figure 3.3 (for example, function 'Sum' is related to a filter B whose coefficients are all 1). The only difference is that the parameters of a convolution layer evolves through the calibration process of the network, while the coefficients of a pooling layer are fixed.

C.0.3 Hyperparameters

We list below the hyperparameters used in the different networks. A grid search has been used to determine which et of hyperparameters lead to the best performance. Table C.2 lists the range of values on which the optimization has been done. The Embedding dimension denotes the dimension N after reduction of the vocabulary V. Batch size is a parameter used in the Batch Gradient Descent (used in the algorithm of optimization of the weights of the networks), see [19]. The number of hidden layers refer to the final multilayer perceptron block used at the output of LSTM and CNN networks. A separate Table (see Table C.3 is devoted to the parameters involved in the embedding part of the procedure.

Hyperparameter	Grid
Embedding dimension	32 to 126
n-grams characters for embedding	1 to 5
Batch size	8 to 256
Number of hidden layers	1 to 5
Number of neurons in each layer	450 to 50
Number of filters for the CNN	32 to 1024
Size of the filters	1 to 5

TABLE C.2 – Range of values on which the hyperparameters have been optimized.

Training algorithm	CBOW
sample	1e-3
embedding dimension	100
window context	3
min word count	10
N-grams	1-2-3
Minimum length of char n-grams	1
Maximum length of char n-grams	4

TABLE C.3 – Hyper-parameters of the Embedding Matrix used in the real data analysis.

All the codes are available at the following url :

`https://github.com/isaaccs/Insurance-reports-through-deep-neural-networks`.

Annexe D

Chapitre 4

D.0.1 Duplication algorithm to correct censoring

In this section, we describe the duplication process used to correct the selection bias caused by the censoring, if we only consider the set of open claims. The core idea is the same as in the weighting procedure used in censored regression to correct the presence of censoring, see [220], [197], [199] or [136] for example. Under the assumption that C is independent from all the other variables $(A, T, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{Z})$, for any function ϕ with finite expectation,

$$E\left[\frac{\delta\phi(\delta A, Y, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{Z})}{S_C(Y)}\right] = E\left[\phi(A, T, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{Z})\right], \tag{D.1}$$

where $S_C(t) = \mathbb{P}(C \geq t)$ is the survival function of the censoring. Hence, applying the weight $W_{i,n}^* = \delta_i S_C(Y_i)^{-1}/n$ is a way to correct the bias caused by the censoring, in the sense that all quantities $\sum_{i=1}^n W_{i,n}^* \phi(\delta_i A_i, Y, \mathbf{X}_i^{(1)}, \mathbf{X}_i^{(2)}, \mathbf{Z}_i))$ almost surely converge towards the appropriate limit $E\left[\phi(A, T, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{Z})\right]$ by the Strong Law of Large Numbers. Since S_C is unknown, it is estimated through the Kaplan-Meier estimator (see [112]) of the censoring survival function, say $\widehat{S}_C$. Hence, an estimation of the ideal weight $W_{i,n}^*$ can be taken as

$$W_{i,n} = \frac{\delta_i}{n\widehat{S}_C(Y_i)}. \tag{D.2}$$

The information on the duration of the open claim is integrated in the weight (D.2), since the computation of $\widehat{S}_C$ relies on both open and closed claims. These weight may be interpreted as means to give more importance in the sample to observations that are under-represented, see also [**?**]. This leads to the following duplication procedure, which is equivalent to considering weighted sums instead of classical empirical means.

Algorithm 7 Duplication algorithm for censored observations

Require:

- a dataset $(\delta_i A_i, \mathbf{Y}_i, \delta_i, \mathbf{X}_i)_{1\leq i\leq n}$, where $\mathbf{X}_i = (\mathbf{X}_i^{(1)}, \mathbf{X}_i^{(2)}, \mathbf{Z}_i)$.
- a list of weights $W_{i,n}$ computed from the data accordingly to (D.2), and let $w = \min_{1\leq i\leq n} W_{i,n}$.

Set $n' = 1$.
for $i \leftarrow 1$ **to** n **do**

1. If $\delta_i = 0$, go to step $i + 1$.
2. Else :
   - Let $k = \lfloor W_{i,n}/w \rfloor$, where $\lfloor x \rfloor$ denotes the integer part of x.
   - Define, for $j \in \{1, ..., k\}$ $A'_{n'-1+j} = \delta_i A_i$, $Y'_{n'-1+j} = Y_i$, $\mathbf{X}_{n'-1+i} = \mathbf{X}_i$.
   - $n' \leftarrow n' + k$.

end for

return The duplicated dataset : $(A'_i, Y'_i, \mathbf{X}'_i)_{1\leq i\leq n'}$.

D.0.2 Hyperparameters

Table des figures

1.1 Comparaison entre Solvabilité I et Solvabilité II 20
1.2 Les trois piliers de Solvabilité II 21
1.3 Triangle des paiements cumulés 27
1.4 Densités des lois de Gumbel et de Fréchet pour différentes valeurs du paramètre de queue et comparaison des densités des lois de reverse Weibull, Fréchet et Gumbel. . 37
1.5 La méthode P.O.T avec un seuil à 6 39
1.6 Comparaison de l'erreur d'estimation en fonction de la méthode. 44
1.7 L'espace des paramètres. Dans le cas ou notre espace est un bol, nous avons un minimum local qui est minimum global. Avec la selle de cheval, nous avons un minimum local qui n'est pas minimum global. Pour la gouttière, on a plusieurs solutions pour un même minimum. 52
1.8 L'algorithme CART 54
1.9 Exemple d'un Tree Bagging avec trois arbres 55
1.10 Random Forest : processus de construction et d'assemblage 56
1.11 Random Forest : processus de construction et d'assemblage 57
1.12 Représentation d'un neurone simple. 59
1.13 Représentation d'un réseaux de neurones et de ses différentes couches. Chaque cercle est un neurone. Chaque rectangle est une couche composée d'un ou plusieurs neurones 59
1.14 Représentation des étapes d'un réseau de neurones. De l'étape forward à la backward. 60
1.15 Représentation du Dropout. Image de l'article "Dropout : a simple way to prevent neural networks from overfitting ."[194] 62
1.16 Un exemple d'une couche de convolution. Chaque coefficient $c_{i,j}$ de la matrice de sortie est obtenue par combinaisons linéaires des coefficients $a_{i,j}$ dans la sous matrice de taille $f \times f$ de la matrice d'entrée. Le code couleur montre quel sous matrice est utilisée pour obtenir le coefficient correspond de la matrice de sortie C. 63
1.17 Un exemple d'une couche de Pooling. Chaque fonction est appliquée à une sous matrice de taille 2x2 64
1.18 Un exemple d'une couche de Padding. 64
1.19 Un RNN déroulé 65
1.20 Une cellule de RNN 66
1.21 La fonction $tanh$ et ses composées 66

1.22 Illustration de problème de mémoire du RNN 67
1.23 Comparaison d'une cellule d'un simple RNN vs une cellule de LSTM. 68
1.24 Notation du LSTM 68
1.25 Représentation de la flèche de contexte qui fait le lien entre l'information précédente, l'actuelle et la future. 68
1.26 Représentation d'une porte à l'aide d'une fonction sigmoïde 69
1.27 Comparaison du F1 Score en fonction de la taille du Batch 74

2.1 Les différents types de variables 82
2.2 À gauche une forme de scaling où la variable sera comprise entre 0 et 1. À droite la formule de la standardisation ou μ est la moyenne de la variable et σ sa variance . . 83
2.3 Représentation de la construction de la matrice d'Embedding selon Mikolov CBOW 85
2.4 Boxplot 88
2.5 Boxplot avec la distribution 89
2.6 Boxplot et histogramme avec et sans valeurs aberrantes 90
2.7 Matrice de corrélation des montants. 92
2.8 QQ-plot des observations 93
2.9 Les estimateurs pour la période de 2002 à 2016 94
2.10 MEF 95
2.11 Graphique de Hill 96
2.12 Zoom du graphique de Hill 96
2.13 GPD 98
2.14 Taux d'inflation appliqué à nos montants 99
2.15 Évolution du quantile 2,5% avec inflation et sans inflation 101
2.16 Risque d'une sur-pondération sur la méthode POT 102
2.17 Fonction de répartition avant pondération 104
2.18 Fonction de répartition avant pondération 104
2.19 Les estimateurs 105
2.21 GPD 105
2.20 Estimation du seuil des valeurs extrêmes à travers deux outils graphiques, la MEF et le Hill Plot. Les deux semblent indiquer un seuil aux alentours des 25000€. 106

3.1 Left-hand side : representation of a simple neural network (multilayer perceptron). Each unit of the network (represented by a circle), is called a neuron (a neuron is represented on the right-hand side). $X^{(j)}$ represents the $j-$th component of the covariate vector $\mathbf{X}$. 112
3.2 Summary of the embedding matrix determination. 117

3.3 An example of a convolution layer. Convolution Layer. Each coefficient $c_{i,j}$ of the output matrix is obtained by a linear combinations of coefficients $a_{i,j}$ in a $f \times f$ square of the input matrix. The color code shows which coefficients of the input matrix have been used to compute the corresponding coefficient of C. 119

3.4 A unrolled recurrent neural network and the unfolding representation in time of the computation involved in its forward computation. 120

3.5 Synthetic description of a block of a Long Short Term Memory network. 120

3.6 LSTM notations. 121

3.7 Representation of the CNN network used in the real data analysis. 124

3.8 Representation of the LSTM network used in the real data analysis. The output of the network is followed by a multilayer perceptron. 124

4.1 Left-hand side : representation of a simple neural network. Each unit of the network (represented by a circle), is called a neuron (a neuron is represented on the right-hand side). $X^{(j)}$ represents the $j-$th component of the covariate vector $\mathbf{X}$. 138

4.2 Synthetic description of a block of a Long Short Term Memory network. 139

4.3 LSTM and Bi-LSTM notations 139

4.4 Short description of a Bi-LSTM network. Each block A has the same structure as in Figure 4.2. 140

4.5 Evolution of the disability rate (DFP) of 4 individuals through time. The label "extreme" indicates a severe claim, "normal" indicates a standard claim. 142

4.6 Hill plot for the variable L. On the top, value of the threshold ; on the bottom, corresponding rank of the observation in the sample. 143

4.7 Summary of the whole procedure. The yellow box corresponds to the prediction model on T. The blue box represents the different neural network structures used to perform feature extraction. The orange box gathers these two inputs and combines a prediction of a severity (using different machine learning techniques) with a model to predict the cost. 143

4.8 On the top left corner, normalized confusion matrice for our neural network at the opening of the claim, on the top right, at the end of the claim. Below, the same timestamp but for our GLM 145

4.9 Relative comparison of claims models against the expert's predictions in the first 4 years. 146

4.10 Relative comparison of extreme claims models against the expert's predictions in the first 4 years. 146

A.1 Les différentes étapes du LSTM 156

B.1 Les estimateurs 157

B.2 Graphique de Hill 157

B.3 GPD 158

B.4 Les estimateurs 158

B.5 Graphique de Hill 158
B.6 GPD 159
B.7 Les estimateurs 159
B.8 Graphique de Hill 159
B.9 GPD 160

C.1 Zero-padding 162
C.2 Example of a Pooling Layer : each function is applied to a moving square of input coefficients of size 2x2. 162

Liste des tableaux

1.1 Distribution de la durée de vie du sinistre en fonction de la charge. La corrélation entre la durée de vie et le coût d'un sinistre corporel semble indiquer que plus un sinistre dure plus celui-ci aura une charge importante. 22

1.2 Quelques familles exponentielles 49

1.3 Fonctions de lien canonique. 50

1.4 Tableau récapitulatif des avantages et inconvénients de chacune des méthodes . . . 53

1.5 Les fonctions d'activation les plus utilisées 61

2.1 Seuils avec la distance relative 87

2.2 Répartition du nombre de victimes par sinistre 91

2.3 Statistiques de la GPD 97

2.4 Quantiles 92,5%, 95%, 97,5% par année de clôture 99

2.5 Quantiles 2,5% par année de clôture via régression 100

2.6 Quantiles 2,5% par année de clôture via régression inflaté 100

2.7 Pente courte période 101

2.8 Statistiques des clos et des en cours 103

2.9 Statistiques de la base suite à Kaplan-Meier 105

3.1 Empirical statistics on the variable T, before and after correction by Kaplan-Meier weights ("after KM"). The category "Extreme claims" corresponds to the situation where $I = 1$ for $x = 3\%$ of the claims, while "Standard claims" refers to the 97% lower part of the distribution of the final amount. 122

3.2 Summarized characteristics of the reports. The dataset is split into the train set (containing the observations used to fit the parameter of the network), the validation set (used to tune the hyper-parameters), and the test set (used to measure the quality of the model). The vocabulary size "with $n-$grams" is the total of $1-$grams, $2-$grams and $3-$grams. 122

3.3 Ranking of the words (translated from French) used in the reports, depending on the category of claims (Extreme corresponds to $I = 1$ and Standard to $I = 0$.) 123

3.4 Hyper-parameters of the recurrent neural network used in the real data analysis. . 124

3.5 Influence of the constrained bagging algorithms of section 3.2.3. For "Classical," no rebalancing algorithm has been used. "Balanced" corresponds to algorithm 5 with an equal proportion of 1 and 0 in each sample. "Randomly" corresponds to algorithm 6 with an equal proportion (in average) of 1 and 0 in each sample. "Lightly" corresponds to algorithm 6 with 10% of 1 (in average) in each sample. In the Random Forest algorithm, we used 200 trees, with a Gini criterion and a maximal depth of 3. In the Gradient Boosting algorithm, 200 trees were used, and a deviance loss with a learning rate of 0.1. 127

3.6 Confidence Interval (CI) of the scoring. The CI for Precision and Recall (columns 4 and 5) are computed using a beta distribution following the methodology of [87]. For bagging (columns 6 to 8), the CI is based on percentile bootstrap following [97]. 128

3.7 Performance of the different methodologies (combined with the different embedding technics). The first column indicates the percentage of 1 in the sample. 129

4.1 General statistics on the database. 141

4.2 F_1-score of the different prediction models for different maturities of claims. . . . 144

4.3 F_1-score of extrems claims of the different prediction models for different maturities of claims. 145

4.4 MSE comparison of our models with $\widehat{T}$ against the mean T, the true T and without T 147

C.1 Typical choices of activation functions. 161

C.2 Range of values on which the hyperparameters have been optimized. 163

C.3 Hyper-parameters of the Embedding Matrix used in the real data analysis. 163

Bibliographie

[1] MS Windows NT assurance automobile – vers une forte hausse de l'indemnisation des sinistres corporels ? https://www.galea-associes.eu/2018/10/assurance-automobile-vers-une-forte-hausse-de-lindemnisation-des-sinistres-corporels/.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow : Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] Charu C Aggarwal and ChengXiang Zhai. *Mining text data*. Springer Science & Business Media, 2012.

[4] Josephine Akosa. Predictive accuracy : A misleading performance measure for highly imbalanced data. In *Proceedings of the SAS Global Forum*, pages 2–5, 2017.

[5] Michael G Akritas et al. The central limit theorem under censoring. *Bernoulli*, 6(6) :1109–1120, 2000.

[6] Neena Aloysius and M Geetha. A review on deep convolutional neural networks. In *2017 International Conference on Communication and Signal Processing (ICCSP)*, pages 0588–0592. IEEE, 2017.

[7] Per K Andersen, Ornulf Borgan, Richard D Gill, and Niels Keiding. *Statistical models based on counting processes*. Springer Science & Business Media, 1998.

[8] Katrien Antonio and Richard Plat. Micro-level stochastic loss reserving for general insurance. *Scandinavian Actuarial Journal*, 2014(7) :649–669, 2014.

[9] Katrien Antonio and Richard Plat. Micro-level stochastic loss reserving for general insurance. *Scandinavian Actuarial Journal*, 2014(7) :649–669, 2014.

[10] Elja Arjas. The claims reserving problem in non-life insurance : Some structural ideas. *ASTIN Bulletin : The Journal of the IAA*, 19(2) :139–152, 1989.

[11] Eugène Astesan. *Université de Paris. Faculté de droit. Les Réserves techniques des sociétés d'assurances contre les accidents d'automobiles, thèse pour le doctorat... présentée... par Eugène Astesan...* R. Pichon et R. Durand-Auzias, 1938.

[12] Yuri Sousa Aurelio, Gustavo Matheus de Almeida, Cristiano Leite de Castro, and Antonio Padua Braga. Learning from imbalanced data sets with weighted cross-entropy function. *Neural Processing Letters*, 50(2) :1937–1949, 2019.

[13] Sophie Balech and C Benavent. Les techniques du nlp pour la recherche en sciences de gestion. 2019.

[14] Heejung Bang and Anastasios A Tsiatis. Estimating medical costs with censored data. *Biometrika*, 87(2) :329–343, 2000.

[15] Gustavo EAPA Batista, Ana LC Bazzan, and Maria Carolina Monard. Balancing training data for automated annotation of keywords : a case study. In *WOB*, pages 10–18, 2003.

[16] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1) :20–29, 2004.

[17] Maximilien Baudry and Christian Y Robert. A machine learning approach for individual claims reserving in insurance. *Applied Stochastic Models in Business and Industry*, 35(5) :1127–1155, 2019.

[18] Jan Beirlant, Yuri Goegebeur, Johan Segers, and Jozef L Teugels. *Statistics of extremes : theory and applications*. John Wiley & Sons, 2006.

[19] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks : Tricks of the trade*, pages 437–478. Springer, 2012.

[20] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2) :157–166, 1994.

[21] S. Benjamin and L. M. Eagles. Reserves in lloyd's and the london market. *Journal of the Institute of Actuaries*, 113(2) :197–256, 1986.

[22] Noureddine Benlagha, Michel Grun&Réhomme, and Olga Vasechko. Les sinistres graves en assurance automobile : Une nouvelle approche par la théorie des valeurs extrêmes. *Revue Modulad*, 47(39), 2009.

[23] Michael W Berry and Malu Castellanos. Survey of text mining. *Computing Reviews*, 45(9) :548, 2004.

[24] Patrice Bertail et al. *Evaluation des risques d'exposition à un contaminant alimentaire : quelques outils statistiques*. INSEE, 2002.

[25] Christopher M. Bishop. Mixture density networks. Technical report, 1994.

[26] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv :1607.04606*, 2016.

[27] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5 :135–146, 2017.

[28] L Bottou and O Bousquet. The tradeoffs of large-scale learning : Optimization for machine learning, 351. 2011.

[29] Léon Bottou et al. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8) :12, 1991.

[30] Axel Brando. Mixture density networks (mdn) for distribution and uncertainty estimation, 2017. Report of the Master's Thesis : Mixture Density Networks for distribution and uncertainty estimation.

[31] Leo Breiman. Bagging predictors. *Machine learning*, 24(2) :123–140, 1996.

[32] Leo Breiman. Random forests. *Machine learning*, 45(1) :5–32, 2001.

[33] Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, Boca Raton, new ed edition, January 1984.

[34] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[35] Hans NE Byström. Extreme value theory and extremely large electricity price changes. *International Review of Economics & Finance*, 14(1) :41–55, 2005.

[36] Julie Carreau and Y. Bengio. A hybrid pareto model for asymmetric fat-tailed data : The univariate case. *Extremes*, 12 :53–76, 03 2009.

[37] Enrique Castillo. *Extreme value theory in engineering*. Elsevier, 2012.

[38] Patricio Cerda, Gaël Varoquaux, and Balázs Kégl. Similarity encoding for learning with dirty categorical variables. *Machine Learning*, June 2018.

[39] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote : synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16 :321–357, 2002.

[40] Tianqi Chen and Carlos Guestrin. Xgboost : A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[41] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, and Yuan Tang. Xgboost : extreme gradient boosting. *R package version 0.4-2*, pages 1–4, 2015.

[42] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv :1601.06733*, 2016.

[43] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation : Encoder-decoder approaches. *arXiv preprint arXiv :1409.1259*, 2014.

[44] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv :1412.3555*, 2014.

[45] Axel Cleeremans, David Servan-Schreiber, and James L McClelland. Finite state automata and simple recurrent networks. *Neural computation*, 1(3) :372–381, 1989.

[46] Stuart Coles, Joanna Bawa, Lesley Trenner, and Pat Dorazio. *An introduction to statistical modeling of extreme values*, volume 208. Springer, 2001.

[47] Jerome T Connor, R Douglas Martin, and Les E Atlas. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, 5(2) :240–254, 1994.

[48] Andrew Cotter, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. *arXiv preprint arXiv :1106.4574*, 2011.

[49] Clément Cousin. Le débat sur le référentiel indicatif de l'indemnisation du préjudice corporel des cours d'appel à l'heure des bases de données. 2017.

[50] David R Cox. Regression models and life-tables. *Journal of the Royal Statistical Society : Series B (Methodological)*, 34(2) :187–202, 1972.

[51] David R Cox. Partial likelihood. *Biometrika*, 62(2) :269–276, 1975.

[52] Harald Cramir. Mathematical methods of statistics. *Princeton U. Press, Princeton*, 500, 1946.

[53] Jonas Crevecoeur and Katrien Antonio. A generalized reserving model : bridging the gap between pricing and individual reserving. *arXiv preprint arXiv :1910.12692*, 2019.

[54] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(1), 2012.

[55] Arnold LM Dekkers, John HJ Einmahl, and Laurens De Haan. A moment estimator for the index of an extreme-value distribution. *The Annals of Statistics*, pages 1833–1855, 1989.

[56] Michel Denuit and Arthur Charpentier. Mathématiques de l'assurance non-vie. 01 2004.

[57] Najmeddine Dhieb, Hakim Ghazzai, Hichem Besbes, and Yehia Massoud. Extreme gradient boosting machine learning algorithm for safe auto insurance operations. In *2019 IEEE International Conference of Vehicular Electronics and Safety (ICVES)*, pages 1–5. IEEE, 2019.

[58] Jean-Pierre Dintilhac et al. Rapport du groupe de travail chargé d'élaborer une nomenclature des préjudices corporels. *Minerva*, 2005.

[59] Pedro Domingos. Metacost : A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164, 1999.

[60] Harris Drucker. Improving regressors using boosting techniques. *Proceedings of the 14th International Conference on Machine Learning*, 08 1997.

[61] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul) :2121–2159, 2011.

[62] Anne Dutfoy, Sylvie Parey, and Nicolas Roche. Multivariate extreme value theory-a tutorial with applications to hydrology and meteorology. *Dependence Modeling*, 1(open-issue), 2014.

[63] Francis Duval and Mathieu Pigeon. Individual loss reserving using a gradient boosting-based approach. *Risks*, 7(3) :79, 2019.

[64] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[65] Martin Ellingsworth and Dan Sullivan. Text mining improves business intelligence and predictive modeling in insurance. *Information Management*, 13(7) :42, 2003.

[66] Paul Embrechts, Claudia Klüppelberg, and Thomas Mikosch. *Modelling extremal events : for insurance and finance*, volume 33. Springer Science & Business Media, 2013.

[67] Sébastien Farkas, Olivier Lopez, and Maud Thomas. Cyber claim analysis through generalized pareto regression trees with applications to insurance pricing and reserving. 2019.

[68] Muhammad Arief Fauzan and Hendri Murfi. The accuracy of xgboost for insurance claim prediction. *Int. J. Adv. Soft Comput. Appl*, 10(2), 2018.

[69] Ronald Aylmer Fisher and Leonard Henry Caleb Tippett. Limiting forms of the frequency distribution of the largest or smallest member of a sample. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 24, pages 180–190. Cambridge University Press, 1928.

[70] Thomas R Fleming and David P Harrington. *Counting processes and survival analysis*, volume 169. John Wiley & Sons, 2011.

[71] Maurice Fréchet. Sur la loi de probabilité de l'écart maximum. *Ann. Soc. Math. Polon.*, 6 :93–116, 1927.

[72] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121(2) :256–285, 1995.

[73] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[74] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[75] Jerome H. Friedman. Greedy function approximation : A gradient boosting machine. *Annals of Statistics*, 29 :1189–1232, 2000.

[76] Janos Galambos. Extreme value theory for applications. In *Extreme value theory and applications*, pages 1–14. Springer, 1994.

[77] Ramazan Gençay, Faruk Selçuk, and Abdurrahman Ulugülyağci. High volatility, thick tails and extreme value theory in value-at-risk estimation. *Insurance : Mathematics and Economics*, 33(2) :337–356, 2003.

[78] Guillaume Gerber, Yohann Le Faou, Olivier Lopez, and Michael Trupin. The impact of churn on client value in health insurance, evaluation using a random forest under random censoring. *Journal of American Statistical Association*, to appear :https ://-doi.org/10.1080/01621459.2020.1764364, 2020.

[79] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1) :3–42, 2006.

[80] Richard Ghueldre and François Vannesson. Doctrine la directive solvency ii la directive 2009/138/ce du parlement européen et du conseil du 25 novembre 2009 sur l'accès aux activités de l'assurance et de la réassurance et leur exercice—solvabilité ii. *Revue Generale du Droit des Assurances*, (3) :611, 2010.

[81] Richard Gill. Large Sample Behaviour of the Product-Limit Estimator on the Whole Line. *The Annals of Statistics*, 11(1) :49 – 58, 1983.

[82] Manfred Gilli et al. An application of extreme value theory for measuring financial risk. *Computational Economics*, 27(2-3) :207–228, 2006.

[83] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, page 580–587, USA, 2014. IEEE Computer Society.

[84] Boris Gnedenko. Sur la distribution limite du terme maximum d'une serie aleatoire. *Annals of mathematics*, pages 423–453, 1943.

[85] Lucien Diégane Gning. *Modèles linéaires généralisés et extensions : utilisation pratique en assurance non-vie*. Éditions Universitaires Européenes, 2012.

[86] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[87] Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. volume 3408, pages 345–359, 04 2005.

[88] Benjamin Graham. Fractional max-pooling. *CoRR*, abs/1412.6071, 2014.

[89] Emil Julius Gumbel. Les valeurs extrêmes des distributions statistiques. In *Annales de l'institut Henri Poincaré*, volume 5, pages 115–158, 1935.

[90] Emil Julius Gumbel. *Statistics of extremes*. Courier Corporation, 2012.

[91] Svend Haastrup and Elja Arjas. Claims reserving in continuous time ; a nonparametric bayesian approach. *ASTIN Bulletin : The Journal of the IAA*, 26(2) :139–164, 1996.

[92] Donatien Hainaut. A neural-network analyzer for mortality forecast. *ASTIN Bulletin : The Journal of the IAA*, 48(2) :481–508, 2018.

[93] Fatma Hamdi. *Apprentissage en distributions déséquilibrées*. PhD thesis, Paris 13, 2012.

[94] Fatma Hamdi, Mustapha Lebbah, and Younès Bennani. Topographic under-sampling for unbalanced distributions. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2010.

[95] Benoit Hamon. https ://www.legifrance.gouv.fr/loda/id/jorftext000028738036/.

[96] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning : data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[97] Tim Hesterberg. What teachers should know about the bootstrap : Resampling in the undergraduate statistics curriculum. *The American Statistician*, 69, 11 2014.

[98] Bruce M Hill. A simple general approach to inference about the tail of a distribution. *The annals of statistics*, pages 1163–1174, 1975.

[99] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8) :1735–1780, 1997.

[100] Arthur E Hoerl and Robert W Kennard. Ridge regression : Biased estimation for nonorthogonal problems. *Technometrics*, 12(1) :55–67, 1970.

[101] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5) :359–366, 1989.

[102] Sergey Ioffe and Christian Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv :1502.03167*, 2015.

[103] Salma Jamal, Stefano Canto, Ross Fernwood, Claudio Giancaterino, Munir Hiabu, Lorenzo Invernizzi, Tetiana Korzhynska, Zachary Martin, and Hong Shen. Machine learning & traditional methods synergy in non-life reserving. *Report of the ASTIN Working Party of the International Actuarial Association. Available online : https ://www. actuaries. org/IAA/Documents/ASTIN/ASTIN_MLTMS% 20Report_SJAMAL. pdf (accessed on 19 July 2019)*, 2018.

[104] Dennis W Jansen and Casper G De Vries. On the frequency of large stock returns : Putting booms and busts into perspective. *The review of economics and statistics*, pages 18–24, 1991.

[105] Arthur F Jenkinson. The frequency distribution of the annual maximum (or minimum) values of meteorological elements. *Quarterly Journal of the Royal Meteorological Society*, 81(348) :158–171, 1955.

[106] Xiaoli Jin and Edward W Jed Frees. Comparing micro-and macro-level loss reserving models. *Presentation at ARIA*, 2013.

[107] Michael I Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Artificial neural networks : concept learning*, pages 112–127. 1990.

[108] Michael I Jordan. Serial order : A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier, 1997.

[109] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip : Compressing text classification models. *arXiv preprint arXiv :1612.03651*, 2016.

[110] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv :1607.01759*, 2016.

[111] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv :1404.2188*, 2014.

[112] Edward L Kaplan and Paul Meier. Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282) :457–481, 1958.

[113] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv :1408.5882*, 2014.

[114] Douglas M Kline and Victor L Berardi. Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing & Applications*, 14(4) :310–318, 2005.

[115] Inna Kolyshkina and Marcel van Rooyen. Text mining for insurance claim cost prediction. In *Data Mining*, pages 192–202. Springer, 2006.

[116] Sotiris B Kotsiantis. Bagging and boosting variants for handling classifications problems : a survey. *The Knowledge Engineering Review*, 29(1) :78–100, 2014.

[117] György Kovács. An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. *Applied Soft Computing*, 83 :105662, 2019. (IF-2019=4.873).

[118] György Kovács. smote-variants : a python implementation of 85 minority oversampling techniques. *Neurocomputing*, 366 :352–354, 2019. (IF-2019=4.07).

[119] Erhard Kremer. A class of autoregressive models for predicting the final claims amount. *Insurance : Mathematics and Economics*, 3(2) :111–119, 1984.

[120] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[121] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[122] Kevin Kuo. Deeptriangle : A deep learning approach to loss reserving. *Risks*, 7(3) :97, 2019.

[123] Pierre Simon de Laplace. Deuxiéme supplément à la théorie analytique des probabilités (paris : Courcier). 1818.

[124] Christian Roholte Larsen et al. An individual claims reserving model. *Astin Bulletin*, 37(1) :113–132, 2007.

[125] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition : A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1) :98–113, 1997.

[126] Yohann Le Faou. *Contributions to the modeling of duration data under censoring : application to the study of health insurance contract churn.* Theses, Sorbonne Université, October 2019.

[127] Y LeCun, J Denker, D Henderson, R Howard, W Hubbard, and L Jacke. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 1990.

[128] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4) :541–551, 1989.

[129] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[130] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks : Tricks of the trade*, pages 9–48. Springer, 2012.

[131] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT : a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4) :1234–1240, 02 2020.

[132] Simon Lee, S Lin, and K Antonio. Delta boosting machine and its application in actuarial modeling, 2015.

[133] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[134] Wei-Chao Lin, Chih-Fong Tsai, Ya-Han Hu, and Jing-Shang Jhang. Clustering-based undersampling in class-imbalanced data. *Information Sciences*, 409 :17–26, 2017.

[135] Olivier Lopez. Single-index regression models with right-censored responses. *Journal of Statistical Planning and Inference*, 139(3) :1082–1097, 2009.

[136] Olivier Lopez. A censored copula model for micro-level claim reserving. *Insurance : Mathematics and Economics*, 87 :1–14, 2019.

[137] Olivier Lopez, Xavier Milhaud, and Pierre-E Thérond. A tree-based algorithm adapted to microlevel reserving and long development claims. *ASTIN Bulletin : The Journal of the IAA*, 49(3) :741–762, 2019.

[138] Olivier Lopez, Xavier Milhaud, Pierre-E Thérond, et al. Tree-based censored regression with applications in insurance. *Electronic journal of statistics*, 10(2) :2685–2716, 2016.

[139] Thomas Mack. Distribution-free calculation of the standard error of chain ladder reserve estimates. *ASTIN Bulletin*, pages 213–225, 1993.

[140] Thomas Mack. The standard error of chain ladder reserve estimates : Recursive calculation and inclusion of a tail factor. *ASTIN Bulletin : The Journal of the IAA*, 29(2) :361–366, 1999.

[141] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv :1804.07612*, 2018.

[142] P. McCullagh and J.A. Nelder. *Generalized Linear Models, Second Edition*. Chapman & Hall/-CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1989.

[143] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4) :115–133, 1943.

[144] Alexander J McNeil. Estimating the tails of loss severity distributions using extreme value theory. *ASTIN Bulletin : The Journal of the IAA*, 27(1) :117–137, 1997.

[145] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[146] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5528–5531. IEEE, 2011.

[147] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[148] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239. IEEE, 2012.

[149] Peter Mulquiney. Artificial neural networks in insurance loss reserving. In *9th Joint International Conference on Information Sciences (JCIS-06)*. Atlantis Press, 2006.

[150] J.A. Nelder and R.J. Verrall. Credibility theory and generalized linear models. *ASTIN Bulletin*, 27(1) :71–82, 1997.

[151] Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1) :125–161, 2013.

[152] Ragnar Norberg. Prediction of outstanding liabilities in non-life insurance 1. *ASTIN Bulletin : The Journal of the IAA*, 23(1) :95–115, 1993.

[153] Ragnar Norberg. Prediction of outstanding liabilities ii. model variations and extensions. *ASTIN Bulletin*, 29(1) :5–25, 1999.

[154] Ragnar Norberg. Prediction of outstanding liabilities ii. model variations and extensions. *ASTIN Bulletin : The Journal of the IAA*, 29(1) :5–25, 1999.

[155] Sankar K Pal and Sushmita Mitra. Multilayer perceptron, fuzzy sets, classifiaction. 1992.

[156] Sankaran Panchapagesan, Ming Sun, Aparna Khare, Spyros Matsoukas, Arindam Mandal, Björn Hoffmeister, and Shiv Vitaladevuni. Multi-task learning and weighted cross-entropy for dnn-based keyword spotting. In *Interspeech*, volume 9, pages 760–764, 2016.

[157] Christian Partrat, Eric Lecœur, Jean-Marie Nessi, Ecaterina Nisipasu, and Olivier Reiz. Provisionnement technique en assurance non-vie, perspectives actuarielles modernes. *Economica*, 2007.

[158] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

[159] Barak A Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2) :263–269, 1989.

[160] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830, 2011.

[161] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove : Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[162] Jessica Pesantez-Narvaez, Montserrat Guillen, and Manuela Alcañiz. Predicting motor insurance claims using telematics data—xgboost versus logistic regression. *Risks*, 7(2) :70, 2019.

[163] James Pickands III et al. Statistical inference using extreme order statistics. *the Annals of Statistics*, 3(1) :119–131, 1975.

[164] Mathieu Pigeon, Katrien Antonio, and Michel Denuit. Individual loss reserving using paid–incurred data. *Insurance : Mathematics and Economics*, 58 :121–131, 2014.

[165] F Planchet. Utilisation de la théorie des valeurs extrêmes dans un contexte de solvabilité ii, 2007.

[166] Frédéric PLANCHET. Tables de mortalité d'expérience pour des portefeuilles de rentiers. *note méthodologique de l'Institut des Actuaires*, 2005.

[167] AC pour l'ANADAVI. Gazette du palais-droit du dommage corporel : revue de jurisprudences. 2013.

[168] Christophe Quézel-Ambrunaz and Manon Viglino. Préjudice d'affection et deuil pathologique : illustration de la perfectibilité de la nomenclature des postes de préjudice. 2019.

[169] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

[170] SI Resnick, H Dress, and L De Haan. How to make a hill plot. *OR & IE University*, 1998.

[171] Sidney Resnick and Cătălin Stărică. Asymptotic behavior of hill's estimator for autoregressive data. *Communications in statistics. Stochastic models*, 13(4) :703–721, 1997.

[172] Sidney I Resnick. *Extreme values, regular variation and point processes*. Springer, 2013.

[173] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[174] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv :1411.2738*, 2014.

[175] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[176] Frank Rosenblatt. The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386, 1958.

[177] Andrea Rotnitzky and James M Robins. Inverse probability weighting in survival analysis. *Wiley StatsRef : Statistics Reference Online*, 2014.

[178] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv :1609.04747*, 2016.

[179] Magali Ruimy. Élaboration d'un véhiculier en assurance automobile.

[180] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[181] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088) :533–536, 1986.

[182] Isaac Cohen Sabban, Olivier Lopez, and Yann Mercuzot. Automatic analysis of insurance reports through deep neural networks to identify severe claims. *Annals of Actuarial Science*, pages 1–26, 2020.

[183] Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014.

[184] Aditya Rizki Saputro, Hendri Murfi, and Siti Nurrohmah. Analysis of deep neural networks for automobile insurance claim prediction. In *International Conference on Data Mining and Big Data*, pages 114–123. Springer, 2019.

[185] Glen A Satten and Somnath Datta. The kaplan–meier estimator as an inverse-probability-of-censoring weighted average. *The American Statistician*, 55(3) :207–210, 2001.

[186] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2) :197–227, 1990.

[187] Magda Schiegl. A model study about the applicability of the chain ladder method. *Scandinavian Actuarial Journal*, 2015(6) :482–499, 2015.

[188] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3, 2003.

[189] Nikolai Vasil'evich Smirnov. Uber die verteilung des allgemeinen gliedes in der variationsreihe. *Metron*, 12 :59–81, 1935.

[190] Nikolaĭ Vasilévich Smirnov. Limit distributions for the terms of a variational series. Am. math. society, 1952.

[191] Richard L Smith. Maximum likelihood estimation in a class of nonregular cases. *Biometrika*, 72(1) :67–90, 1985.

[192] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc : a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer, 2006.

[193] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity : The all convolutional net, 2014.

[194] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1) :1929–1958, 2014.

[195] Erwin Straub and Swiss Association of Actuaries (Zürich). *Non-life insurance mathematics*. Number 517/S91n. Springer, 1988.

[196] W. Stute and J.-L. Wang. The strong law under random censorship. *Ann. Statist.*, 21(3) :1591–1607, 09 1993.

[197] Winfried Stute. The central limit theorem under random censorship. *The Annals of Statistics*, pages 422–439, 1995.

[198] Winfried Stute. Distributional convergence under random censorship when covariables are present. *Scandinavian journal of statistics*, pages 461–471, 1996.

[199] Winfried Stute. Nonlinear censored regression. *Statistica Sinica*, pages 1089–1102, 1999.

[200] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.

[201] Nguyen Thai-Nghe, Zeno Gantner, and Lars Schmidt-Thieme. Cost-sensitive learning methods for imbalanced data. In *The 2010 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2010.

[202] Maud Thomas and Holger Rootzén. Prédiction d'épidémies de grippe extrêmes.

[203] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society : Series B (Methodological)*, 58(1) :267–288, 1996.

[204] Ivan Tomek et al. Two modifications of cnn. 1976.

[205] Charles Tremblay. Prédire les sinistres graves en assurance : les apports de l'apprentissage statistique aux modèles linéaires.

[206] Anastasios Tsiatis. *Semiparametric theory and missing data*. Springer Science & Business Media, 2007.

[207] Mark J Van der Laan, MJ Laan, and James M Robins. *Unified methods for censored longitudinal data and causality*. Springer Science & Business Media, 2003.

[208] Ingrid Van Keilegom and Michael G. Akritas. Transfer of tail information in censored regression models. *Ann. Statist.*, 27(5) :1745–1784, 10 1999.

[209] Naufal Verdikha, Teguh Adji, and Adhistya Permanasari. Study of undersampling method : Instance hardness threshold with various estimators for hate speech classification. *IJITEE (International Journal of Information Technology and Electrical Engineering)*, 2, 12 2018.

[210] Juanjuan Wang, Mantao Xu, Hui Wang, and Jiwu Zhang. Classification of imbalanced data by using the smote algorithm and locally linear embedding. In *2006 8th international Conference on Signal Processing*, volume 3. IEEE, 2006.

[211] Lee-Jen Wei. The accelerated failure time model : a useful alternative to the cox regression model in survival analysis. *Statistics in medicine*, 11(14-15) :1871–1879, 1992.

[212] TST Wong, WK Li, et al. A note on the estimation of extreme value distributions using maximum product of spacings. In *Time Series and Related Topics*, pages 272–283. Institute of Mathematical Statistics, 2006.

[213] Julien Worms and Rym Worms. Extreme value statistics for censored data with heavy tails under competing risks. *Metrika*, 81(7) :849–889, 2018.

[214] Qingyao Wu, Yunming Ye, Haijun Zhang, Michael K. Ng, and Shen-Shyang Ho. Forestexter : An efficient random forest algorithm for imbalanced text categorization. *Knowledge-Based Systems*, 67 :105 – 116, 2014.

[215] Diethelm Wuertz, Tobias Setz, Yohan Chalabi, Maintainer Tobias Setz, and Suggests RUnit. Package 'fextremes'. 2009.

[216] Mario V Wüthrich. Machine learning in individual claims reserving. *Scandinavian Actuarial Journal*, 2018(6) :465–480, 2018.

[217] Mario V Wüthrich. Neural networks applied to chain–ladder reserving. *European Actuarial Journal*, 8(2) :407–436, 2018.

[218] Mario V Wüthrich. Bias regularization in neural network models for general insurance pricing. *European Actuarial Journal*, pages 1–24, 2019.

[219] Mai Zhou. M-estimation in censored linear models. *Biometrika*, 79(4) :837–841, 1992.

[220] Mai Zhou. M-estimation in censored linear models. *Biometrika*, 79(4) :837–841, 1992.

[221] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society : series B (statistical methodology)*, 67(2) :301–320, 2005.