UNIVERSITÉ DU QUÉBEC À MONTRÉAL

GRADIENT BOOSTING TECHNIQUES FOR INDIVIDUAL LOSS RESERVING IN NON-LIFE INSURANCE

DISSERTATION

PRESENTED

AS PARTIAL REQUIREMENT

TO THE MASTERS IN MATHEMATICS

BY

FRANCIS DUVAL

JUNE 2019

UNIVERSITÉ DU QUÉBEC À MONTRÉAL

TECHNIQUES DE GRADIENT BOOSTING POUR LA MODÉLISATION DES RÉSERVES INDIVIDUELLES EN ASSURANCE NON-VIE

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAITRISE EN MATHÉMATIQUES

PAR

FRANCIS DUVAL

JUIN 2019

REMERCIEMENTS

Je souhaite remercier en tout premier lieu mon directeur de recherche, Mathieu Pigeon, qui m'a soutenu et guidé durant mes deux années à la maitrise, plus particulièrement durant la rédaction de ce mémoire. Grâce à lui, il m'a été possible de soumettre mon premier article scientifique. Je salue entre autres sa grande pédagogie, son assiduité ainsi que son professionnalisme. Je remercie également Desjardins Assurances Générales, qui a fournit les données sans lesquelles ce projet ne se serait pas matérialisé. Plus particulièrement, je tiens à remercier Danaïl Davidov, mon superviseur de stage au cours duquel ce projet a débuté. Il m'a donné les outils nécessaires à l'élaboration des modèles, entre autres en m'initiant à l'apprentissage automatique. Finalement, je tiens à remercier ma famille qui m'a soutenu tout au long de mes études à la maitrise.

AVANT-PROPOS

L'idée du projet de recherche présenté dans ce mémoire a vu le jour lors d'un stage de recherche au sein de Desjardins Assurances Générales achevé à l'été 2017, vers le commencement de ma maitrise en sciences actuarielles. Ce stage a été financé à parts égales par Desjardins Assurances Générales et par Mitacs, un organisme à but non lucratif visant à stimuler l'innovation et la recherche en établissant des partenariats entre le milieu universitaire et l'industrie. Lors de ce stage, le mandat de trouver un algorithme permettant de prédire les montants futurs à payer pour des réclamations passées m'a été donné. À ce moment, les techniques de modélisation des réserves individuelles présentes dans la littérature étaient pour la plupart des méthodes paramétriques. Puisque l'apprentissage statistique était alors un champ d'étude qui produisait des résultats prometteurs dans divers domaines, j'ai développé un algorithme basé sur une méthode non paramétrique d'apprentissage automatique appelée *gradient boosting*, qui a été implanté avec succès dans les systèmes de Desjardins Assurances Générales.

L'algorithme de *gradient boosting* utile au développement du modèle est une boite noire et à l'automne 2017, beaucoup de détails concernant le fonctionnement de cet algorithme m'échappaient. Dans le cadre d'un cours de maitrise, j'ai donc décidé d'apprendre en détail le fonctionnement de cet algorithme, ce qui a abouti à un document expliquant le fonctionnement du *gradient boosting*, de l'algorithme *random forest* ainsi que des arbres de décision, trois techniques connexes appartenant à la famille des algorithmes d'apprentissage machine. Ceci m'a permis d'acquérir une connaissance approfondie du modèle de prédiction des réserves que j'ai développé, ce qui m'a par la suite permis de le perfectionner.

L'été dernier, j'ai eu la chance de présenter les résultats partiels de ce projet au *Joint Statistical Meeting* tenu à Vancouver, le plus grand rassemblement de statisticiens en Amérique du Nord. Il s'agissait d'une courte présentation de cinq minutes suive d'une séance d'affichage d'une durée d'une heure.

Ce projet a également engendré un article qui a été soumis pour publication au moment d'écrire ces lignes. Cet article a été coécrit par moi et Mathieu Pigeon, mon directeur des travaux de recherche à la maitrise.

Finalement, les résultats finaux de mes travaux effectués à la maitrise ont été présentés au *Congrès annuel 2019 à Calgary* de la Société statistique du Canada.

CONTENTS

LIST OF TABLES vi
LIST OF FIGURES vii
RÉSUMÉ viii
ABSTRACT ix
INTRODUCTION 1
CHAPTER I RESERVING PROBLEM SETTING 4
CHAPTER II CLASSICAL MODELS FOR LOSS RESERVING 9
2.1 Chain-ladder Algorithm and Mack's Model 9
2.2 Generalized Linear Models 12
CHAPTER III STATISTICAL LEARNING AND GRADIENT BOOSTING 15
3.1 Supervised Machine Learning 16
3.2 Gradient Boosting 19
3.2.1 TreeBoost 24
3.2.2 Regularization 28
3.3 Gradient Boosting for Loss Reserving 29
CHAPTER IV ANALYSIS 31
4.1 Data 31
4.2 Covariates 35
4.3 Training of *XGBoost* models 37
4.4 Results 45
CONCLUSION 58
BIBLIOGRAPHY 60

LIST OF TABLES

Table Page

4.1.1 Comparison of complete set of claimants ($\mathcal{S}$) and set of claimants from accident years 2004-2010 ($\mathcal{S}_7$). % of open claims is on December 31st 2016. 36

4.1.2 Details about training and validation datasets. 36

4.2.1 Covariates used in the models. 37

4.3.1 Main specifications of XGBoost models. Unless otherwise stated, we have $k \in \mathcal{T}$. 44

4.4.1 Training incremental run-off triangle (in \$100,000). 45

4.4.2 Validation incremental run-off triangle (in \$100,000). 46

4.4.3 Prediction results for collective approaches. 48

4.4.4 Prediction results for individual generalized linear models using covariates. 52

4.4.5 Prediction results for individual approaches using covariates. . . . 53

LIST OF FIGURES

Figure Page

1.0.1 Development of the claim k. 5

4.1.1 Number of claimants per claim. 33

4.1.2 Distribution of final incurred by accident years (on a base 10 log scale). The first quartile is equal to the minimum for all accident years since many claims close at zero. The average incurred for each accident year is represented by a dot. 34

4.1.3 Status of claims on December 31st 2016. 35

4.3.1 Status of claims on December 31st 2010. 38

4.4.1 Comparison of predictive distributions for collective models. The observed reserve amount is represented by the vertical dashed line. 48

4.4.2 Estimated parameters for quasi-Poisson individual GLM. The black dots correspond to the out-of-sample estimates, as the grey dot are the in-sample estimates. 51

4.4.3 Predictive distributions for in-sample and out-of-sample individual GLM with covariates. 52

4.4.4 Predictive distributions for *XGBoost* **Model A**, **C1**, **C2**, and **C3**. 54

4.4.5 Predictive distributions for *XGBoost* **Model A**, **D1**, **D2**, **D3** and **D4**. 55

4.4.6 Comparison of predictive distributions for **Model E** and **Model C3**. The observed total paid amount is represented by the vertical dashed line. 56

RÉSUMÉ

La modélisation fondée sur des données est l'un des sujets de recherche qui pose le plus de défis dans la science actuarielle pour le provisionnement et l'évaluation du risque. La plupart des analyses sont basées sur des données agrégées, mais il est clair aujourd'hui que cette approche ne dit pas tout sur une réclamation et ne décrit pas précisément son évolution. Les approches d'apprentissage statistique en général, et les algorithmes de *gradient boosting* en particulier, offrent un ensemble d'outils qui pourraient aider à évaluer les réserves dans un cadre individuel. Dans ce mémoire, nous comparons certaines techniques agrégées traditionnelles (au niveau du portefeuille) avec des modèles individuels (au niveau de la réclamation) basés à la fois sur des modèles paramétriques et sur des algorithmes de *gradient boosting*. Ces modèles individuels utilisent de l'information sur chacun des paiements effectués pour chacune des réclamations du portefeuille, ainsi que sur les caractéristiques de l'assuré. Nous fournissons un exemple basé sur un ensemble de données détaillées provenant d'une compagnie d'assurance non-vie et nous discutons de certains points liés aux applications pratiques.

Mots-clé: assurance non-vie, provisionnement, modélisation prédictive, modèles individuels, gradient boosting

ABSTRACT

Modeling based on data information is one of the most challenging research topics in actuarial science for loss reserving and risk valuation. Most of the analyzes are based on aggregate data but nowadays it is clear that this approach does not tell the whole story about a claim and does not describe precisely its development. Statistical learning approaches in general, and gradient boosting algorithms in particular, offer a set of tools that could help to evaluate loss reserves in an individual framework. In this work, we contrast some traditional aggregate techniques (portfolio-level) with individual models (claim-level) based on both parametric models and gradient boosting algorithms. These claim-level models use information about each of the payments made for each of the claims in the portfolio, as well as characteristics of the insured. We provide an explicit example based on a detailed dataset from a property and casualty insurance company and we discuss some points related to practical applications.

Key words: non-life insurance, loss reserving, predictive modeling, individual models, gradient boosting

INTRODUCTION

In its daily practice, a non-life insurance company is subject to a number of solvency constraints, e.g., ORSA guidelines in North America and Solvency II in Europe. More specifically, an actuary must predict, with the highest accuracy, future claims based on past observations. The difference between the total predicted amount and the total of all amounts already paid represents a reserve that the company must set aside. A substantial part of the actuarial literature is devoted to the modeling, the evaluation and the management of this risk (see [WM08] for an overview of existing methods).

Almost all existing models can be divided into two categories depending on the granularity of the underlying dataset: individual, or micro-level, approaches when most information on contracts, claims, payments, etc. has been preserved and collective, or macro-level, approaches when an aggregation to some extent has been made (often on an annual basis). The latter have been widely developed by researchers and successfully applied by practitioners for several decades. The former have been studied for few decades but actual use is very rare despite the many advantages of these methods.

The idea of using an individual model – or a structural stochastic description – for claims dates back to the early 1980's with, among others, [BU80], [HA80] and [NO86]. The latter has proposed an individual model describing the occurrence, the reporting delay and the severity of each accident separately. The idea was followed by the work of [AR89], [NO93A, NO99], [HE94], [JE89] and [HA96]. This period was characterized by very limited computing and memory resources

as well as by the lack of usable data on individual claims. However, we can find some applications in [HA96] and in some more technical documents such as [NO93B] and [KI94].

Since the beginning of the 2000's, several works have been done – mainly in the marked (Poisson) process framework – including the modeling of the dependence using copulas [ZZ10], the use of generalized linear models [LA07], the semi-parametric modeling of certain components [AP14] and [ZZ09], the use of skew-symmetric distributions [PA13], the inclusion of additional information [TM08], etc. Finally, some researchers have focused on the comparisons that can be made between individual and collective approaches, often attempting to answer the question "What is the best approach?" (see [HU15], [HI16] or [CP16] for some examples).

Nowadays, statistical learning techniques are widely used in the field of data analytics and may offer non-parametric solutions to claim reserving. These methods give more freedom to the model and often outperform the accuracy of their parametric counterparts. However, only few non-parametric approaches have been developed using micro-level information. One of them is presented in [WU18], where the number of payments is modeled using regression trees in a discrete time framework. The occurrence of a claim payment is assumed to have a Bernoulli distribution, and the probability is then computed using a regression tree as well as all available characteristics. Other researchers, see [BR17], have also developed a non-parametric approach using a machine learning algorithm known as *Extra-Trees*, an ensemble of many unpruned regression trees, for loss reserving.

In this paper, we propose and analyze an individual model for loss reserving based on an application of a gradient boosting algorithm. Gradient boosting is a machine learning technique, which combines sequentially many "simple" models called *weak*

learners to form a stronger predictor by optimizing some objective function. We apply an algorithm called *XGBoost*, see [CG16], to learn a function to predict the ultimate claim amount of a file using all available information at a given time. This information can be about the claimant as well as the claim itself. We also present and analyze micro-level models belonging to the class of generalized linear models (GLM). Based on a detailed dataset from a property and casualty insurance company, we study some properties and we compare results obtained from various approaches. More specifically, we show that the approach combining the *XGBoost* algorithm and a classical collective model such as the Mack's model, has high predictive power and stability. We also propose a method for dealing with censored data and discuss the presence of dynamic covariates. We believe that the gradient boosting algorithm could be an interesting addition to the range of tools available for actuaries to evaluate the solvency of a portfolio.

In Chapter I, we introduce some notation and we present the context of loss reserving from both collective and individual point of view. In this work, our main objective is to present and to analyze micro-level approaches for loss modeling. We also compare collective and individual approaches. Thus, in Chapter II, we present classical collective models as well as GLM for individual data. In Chapter III, we present individual models based on machine learning methods, focusing on gradient boosting techniques for regression problems. A case study and some numerical analyzes on real data are performed in Chapter IV, and finally, we conclude and present some promising generalizations.

CHAPTER I

RESERVING PROBLEM SETTING

In non-life insurance, a claim always starts by an accident experienced by a policyholder that may lead to financial damages covered by an insurance contract. We call *occurrence point* (T_1) the date on which the accident happens. For some situations (bodily injury liability coverage, accident benefits, third party responsibility liability, etc.), a *reporting delay* is observed between the occurrence point and the notification to the insurer at the *reporting point* (T_2). From T_2, the insurer could observe details about the claim, as well as some information about the insured, and record a first estimation of the final amount called *case estimate*. Once the accident is reported to the insurance company, the claim is usually not settled immediately: the insurer has to investigate, wait for bills, wait for court judgments, etc. At the reporting point T_2, a series of M random payments $P_{t_1}, \ldots, P_{t_M} > 0$ made respectively at times $t_1 < \ldots < t_M$ is therefore triggered, until the claim is closed at the *settlement point* (T_3). It should be noted that it is possible for a claim to close without any payment. All the dates are expressed in number of years since an *ad hoc* starting point noted by τ. Finally, we need a unique index k to distinguish the claims. For instance, $T_1^{(k)}$ is the occurrence date of the claim k, and $t_m^{(k)}$ is the date of the m^{th} payment of this claim. At Figure 1.0.1, we illustrate the development of a claim.

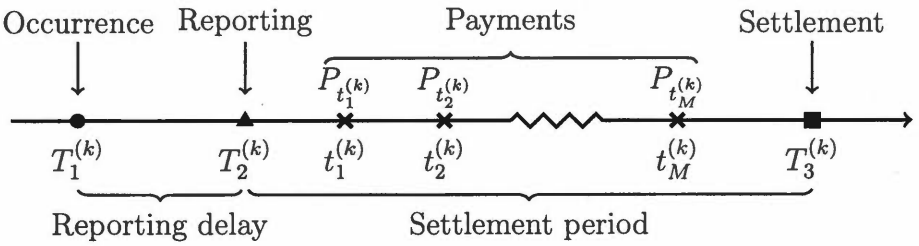

Figure 1.0.1: *Development of the claim k.*

The *evaluation date* t^* is the moment on which the insurance company wants to evaluate the solvency and compute reserves. At this point, a claim can be classified in three categories:

1. If $T_1^{(k)} < t^* < T_2^{(k)}$, the accident has happened but has not yet been reported to the insurer. It is therefore called an *Incurred But Not Reported*, or IBNR, accident. For one of those accidents, the insurer does not have specific information about the accident, but can however use claimant and external information to estimate the reserve.

2. If $T_2^{(k)} < t^* < T_3^{(k)}$, the accident has been reported to the insurer but the claim is still not settled, which means the insurer expects to make additional payments to the insured. It is therefore called a *Reported But Not Settled*, or RBNS, claim. For one of such claims, the history information as well as claimant and external information can be used to estimate the reserve.

3. If $t^* > T_3^{(k)}$, the claim is classified as *Settled*, or S, and the insurer does not expect more payment to be done.

Let $C_t^{(k)}$ be a random variable representing the cumulative paid amount at date t

for claim k and defined by

$$C_t^{(k)} = \begin{cases} 0, & t < T_2^{(k)} \\ \sum_{\{m:t_m^{(k)} \leq t\}} P_{t_m^{(k)}}, & t \geq T_2^{(k)}. \end{cases}$$

At any evaluation date $T_1^{(k)} < t^* < T_3^{(k)}$ and for a claim k, an insurer estimates the cumulative payments amount at the settlement $C_{T_3}^{(k)}$, called *total paid amount*, by $\widehat{C}_{T_3}^{(k)}$ using all useful information available at t^* and denoted by $\mathcal{D}_{t^*}^{(k)}$. The individual reserve for a claim evaluated at t^* is thus given by

$$\widehat{R}_{t^*}^{(k)} = \widehat{C}_{T_3}^{(k)} - C_{t^*}^{(k)}.$$

For the whole portfolio, the total reserve is the aggregation of all individual reserves and is given by

$$\widehat{R}_{t^*} = \sum_{k=1}^{K(t^*)} \widehat{R}_{t^*}^{(k)},$$

where $K(t^*) = |\{k : T_1^{(k)} < t^*\}|$ is the number of claims in the portfolio with evaluation date t^*.

Traditionally, insurance companies aggregate information by accident year and by development year. Claims with accident year i, $i = 1, \ldots, I$, correspond to all the reported accidents that occurred in the i^{th} year after τ, which means all claims k for which $i - 1 < T_1^{(k)} < i$ is verified. For a claim k, a payment made in development year j is a payment made in the j^{th} year after the occurrence $T_1^{(k)}$, namely a payment $P_{t_m^{(k)}}$ for which $j - 1 < t_m^{(k)} - T_1^{(k)} < j$.

Example 1.0.1 *Let τ be January 1^{st} 2004. Claims with accident year 1 are all the reported accidents that occurred between $t = 0$ and $t = 1$, that is to say between January 1^{st} and December 31^{st} 2004, claims with accident year 2 are all reported accidents that occurred between January 1^{st} and December 31^{st} 2005, etc.*

For development years $j = 1, \ldots, J$, we define

$$Y_j^{(k)} = \sum_{m \in \mathcal{S}_j^{(k)}} P_{t_m^{(k)}},$$

where

$$\mathcal{S}_j^{(k)} = \{m : j - 1 < t_m^{(k)} - T_1^{(k)} < j\},$$

as the total paid amount for claim k during year j and we define the corresponding cumulative paid amount as

$$C_j^{(k)} = \sum_{s=1}^{j} Y_s^{(k)}.$$

Collective approaches group every claim having the same accident year together to form the aggregate incremental payment

$$Y_{ij} = \sum_{k \in \{k : i-1 < T_0^{(k)} < i\}} Y_j^{(k)}, \qquad i, j = 1, \ldots, I.$$

Therefore, assuming that $I = J = N$, at the evaluation date $t^* = N$ an insurer owns the information, i.e. the run-off triangle

$$\begin{bmatrix} Y_{11} & Y_{12} & \ldots & Y_{1(N-1)} & Y_{1N} \\ Y_{21} & Y_{22} & \ldots & Y_{2(N-1)} & \\ \vdots & \vdots & \ddots & & \\ Y_{(N-1)1} & Y_{(N-1)2} & & & \\ Y_{N1} & & & & \end{bmatrix}, \tag{1.0.1}$$

where rows represent accident years and columns, development years. It is also possible to write this information in the cumulative form

$$\begin{bmatrix} C_{11} & C_{12} & \ldots & C_{1(N-1)} & C_{1N} \\ C_{21} & C_{22} & \ldots & C_{2(N-1)} & \\ \vdots & \vdots & \ddots & & \\ C_{(N-1)1} & C_{(N-1)2} & & & \\ C_{N1} & & & & \end{bmatrix}, \tag{1.0.2}$$

where $C_{ij} = \sum_{s=1}^{j} Y_{is}$. A prediction at time $t^* = N$ of the reserve is obtained by completing the bottom right part of the triangle in Equation 1.0.1:

$$\widehat{R}_N = \sum_{i=2}^{N} \sum_{j=N+2-i}^{N} \widehat{Y}_{ij}, \tag{1.0.3}$$

where the $\widehat{Y}_{ij}$ are usually predicted (see Section 2) using only the accident year and the development year.

For individual approaches, each cell contains a series of payments, some information about the claims as well as some information about claimants. A prediction of the total reserve amount is given by

$$\widehat{R} = \underbrace{\sum_{i=2}^{N} \sum_{j=N+2-i}^{N} \sum_{k=1}^{K_i^{\text{obs.}}} \widehat{Y}_{ij}^{(k)}}_{\text{RBNS reserve}} + \underbrace{\sum_{i=2}^{N} \sum_{j=N+2-i}^{N} \sum_{k=1}^{\widehat{K}_i - K_i^{\text{obs.}}} \widehat{Y}_{ij}^{(k)}}_{\text{IBNR reserve}}, \tag{1.0.4}$$

where $K_i^{\text{obs.}}$ is the observed number of claims with occurrence year i and the $\widehat{Y}_{ij}^{(k)}$ are now predicted using all available information.

In this work, we focus on estimating the RBNS reserve, which is the first part on the right hand side of Equation 1.0.4, and we forget about the IBNR one.

CHAPTER II

CLASSICAL MODELS FOR LOSS RESERVING

In this section, we briefly describe key approaches for loss reserving in a collective framework. In particular, we introduce the Mack's model (see Subsection 2.1) and the generalized linear models for reserves (see Subsection 2.2). The objective here is not to address the subject in a comprehensive manner: many references already do it perfectly well. We rather want to present the main ideas to allow the reader to read more easily the full paper.

2.1 Chain-ladder Algorithm and Mack's Model

The chain-ladder (CL) algorithm is a non-parametric deterministic reserving method constructed for a cumulative run-off triangle as given by Equation 1.0.2. It is based on two hypotheses:

1. Cumulative payments belonging to different accident years are independent: $\{C_{ij}\}_{j=1}^{N} \perp\!\!\!\perp \{C_{i'j}\}_{j=1}^{N}$ for $i \neq i'$.

2. There exist development factors $\lambda_1, \ldots, \lambda_{N-1}$ such that λ_{j-1} depicts the claim development from the development year $j-1$ to the development year j

$$C_{ij} = \lambda_{j-1} C_{i(j-1)}, \qquad 1 \leq i \leq N, \quad 2 \leq j \leq N.$$

Development factors are estimated using past observations and are usually given by

$$\widehat{\lambda}_j = \frac{\sum_{i=1}^{N-j-1} C_{i(j+1)}}{\sum_{i=1}^{N-j-1} C_{ij}}, \qquad j = 1, \ldots, N-1. \tag{2.1.1}$$

Based on these estimated development factors, it is possible to predict ultimate cumulative payments, namely $\widehat{C}_{2N}, \ldots, \widehat{C}_{NN}$, by developing for each accident year the most recent cumulative payment

$$\widehat{C}_{iN} = (\widehat{\lambda}_{N-i} \times \ldots \times \widehat{\lambda}_{N-1}) \times C_{i(N-i)}, \qquad i = 2, \ldots, N, \tag{2.1.2}$$

and the reserve for accident year i can be predicted by

$$\widehat{R}_i = \widehat{C}_{iN} - C_{i(N-i)}. \tag{2.1.3}$$

Since all occurrence years are independent, the total reserve is obtained by simply adding reserves for all accident years

$$\widehat{R} = \sum_{i=2}^{N} \widehat{R}_i. \tag{2.1.4}$$

Chain-ladder algorithm has a major drawback: it is a deterministic approach and it therefore only gives a point estimate for the reserve. Mack's model [MA93] is a stochastic version of the chain-ladder algorithm and is aiming at estimating prediction error. Mack model relies on the two following hypotheses:

1. Cumulative payments belonging to different accident years are independent random vectors: $\{C_{ij}\}_{j=1}^{N} \perp\!\!\!\perp \{C_{i'j}\}_{j=1}^{N}$ for $i \neq i'$.

2. There exist factors $\lambda_1, \ldots, \lambda_{N-1}$ and variance parameters $\sigma_1^2, \ldots, \sigma_{N-1}^2 > 0$ such that for $1 \leq i \leq N$ and for $2 \leq j \leq N$, we have

$$\mathbb{E}\left[C_{ij} | C_{i(j-1)}\right] = \lambda_{j-1} C_{i(j-1)}$$
$$\mathrm{Var}\left[C_{ij} | C_{i(j-1)}\right] = \sigma_{j-1}^2 C_{i(j-1)}.$$

Variance parameters are estimated using the unbiased estimator

$$\widehat{\sigma}_j^2 = \frac{\sum_{i=1}^{N-j-1} C_{ij}\left(\frac{C_{i(j+1)}}{C_{ij}} - \widehat{\lambda}_j\right)}{N-j-1}, \qquad j = 1, \ldots, N-2.$$

Since the estimation of σ_{N-1}^2 using this formula is impossible, it is generally estimated by extrapolating such that

$$\frac{\widehat{\sigma}_{N-3}^2}{\widehat{\sigma}_{N-2}^2} = \frac{\widehat{\sigma}_{N-2}^2}{\widehat{\sigma}_{N-1}^2},$$

while making sure that $\widehat{\sigma}_{N-3}^2 > \widehat{\sigma}_{N-2}^2$. Hence, $\widehat{\sigma}_{N-1}^2$ is computed using

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

Finally, development factors $\lambda_1, \ldots, \lambda_{N-1}$ are estimated using Equation 2.1.1. As with chain-ladder algorithm, $\widehat{C}_{iN}$ and $\widehat{R}_i$ for $i = 2, \ldots, N$ can be obtained using Equation 2.1.2 and Equation 2.1.3 respectively. Since both chain-ladder and Mack methods use the same development factors, they lead to the same reserve estimates. Mack's model differs from chain-ladder algorithm in that it is possible to compute the variance of the reserves. The variance for the reserve of accident year i is given by

$$\widehat{\mathrm{Var}\left[\widehat{R}_i\right]} = \widehat{C}_{iN}^2 \sum_{k=N-i+1}^{N-1} \frac{\widehat{\sigma}_k^2}{\widehat{\lambda}_k^2}\left(\frac{1}{\widehat{C}_{ik}} + \frac{1}{\sum_{s=1}^{N-k}\widehat{C}_{sk}}\right), \qquad i = 2, \ldots, N$$

and the variance of the total reserve amount is given by

$$\widehat{\mathrm{Var}\left[\widehat{R}\right]} = \sum_{i=2}^{N}\left(\widehat{\mathrm{Var}\left[\widehat{R}_i\right]} + \widehat{C}_{iN}\left(\sum_{s=i+1}^{N}\widehat{C}_{sN}\right)\sum_{k=N-i+1}^{n-1}\frac{2\widehat{\sigma}_k^2/\widehat{\lambda}_k^2}{\sum_{s=1}^{N-k}C_{sk}}\right).$$

Proofs of the formulas and more details are given in [MA93], as well as in classic textbooks such as [WM08]. Since Mack's model does not assume any distribution

for the payments, the predictive distribution of the total reserve can be estimated using a bootstrap approach, such as the one described in [EV02].

Since its creation in 1993, Mack's model has been extensively studied, and several variants and extensions have been developed. Two well known examples are Munich chain-ladder [QM04] and London Chain [BE86] methods. In Munich chain-ladder, paid losses Y_{ij} of the incremental run-off triangle are replaced by quotients of paid and incurred losses Y_{ij}/I_{ij}, which allows to take into consideration correlation between paid an incurred losses. In London Chain method, payments are assumed to have not only a multiplicative trend, but also an additive trend. Therefore, we assume that there exists multiplicative development factors $\lambda_1, \ldots, \lambda_{N-1}$ and additive development factors $\alpha_1, \ldots, \alpha_{N-1}$ such that

$$C_{ij} = \lambda_{j-1} C_{i(j-1)} + \alpha_{j-1}, \qquad 1 \leq i \leq N, \quad 2 \leq j \leq N.$$

Multiplicative and additive development factors are estimated jointly by least square.

2.2 Generalized Linear Models

For a response variable for which the distribution is a member of the linear exponential family, a generalized linear model, or GLM [MN89], is built from:

- a linear predictor $\boldsymbol{x}\boldsymbol{\beta}$, where $\boldsymbol{x}$ is the row vector of predictors and $\boldsymbol{\beta}$ is the column vector of parameters;
- a bijective link function g that describes the relation between the expectation of the response variable Y and the linear predictor, i.e., $g\left(\mathbb{E}[Y|\boldsymbol{x}]\right) = \boldsymbol{x}\boldsymbol{\beta}$; and
- a variance function $\mathcal{V}$ that describes the link between the variance and the

expectation of the response variable Y, i.e., $\text{Var}[Y|\boldsymbol{x}] = \varphi\mathcal{V}(\mathbb{E}[Y|\boldsymbol{x}])$, where φ is a dispersion parameter.

GLM are widely used in many fields including biology and psychology. In actuarial science, they are commonly used for pricing.

In a collective framework, each cell of Triangle 1.0.1 is modeled using

$$g(\mathbb{E}[Y_{ij}|i,j]) = \beta_0 + \alpha_i + \beta_j$$

and

$$\text{Var}[Y_{ij}|i,j] = \varphi\mathcal{V}(\mathbb{E}[Y_{ij}|i,j]),$$

where α_i, $i = 2,3,\ldots,N$ is the accident year effect, β_j, $j = 2,3,\ldots,N$ the development year effect and β_0 is the intercept. Finally, all Y_{ij}'s are assumed to be independent. The prediction for cell in position i, j of the triangle in Equation 1.0.1 is given by

$$\widehat{Y}_{ij} = g^{-1}(\widehat{\beta}_0 + \widehat{\alpha}_i + \widehat{\beta}_j),$$

where estimates of the parameters $\widehat{\beta}_0$, $\widehat{\alpha}_i$ and $\widehat{\beta}_j$ are usually found by maximizing likelihood. We can therefore obtain an estimate of the reserve using Equation 1.0.3. Since we assume each cell of the triangle is the realization of a random variable following a distribution of the linear exponential family, it is not only possible to compute the first and second moments, but also the whole predictive distribution of the reserve.

In the individual framework, in addition to accident and development year, it becomes possible to use specific features of each claim k, and we have

$$\begin{aligned} g\left(\mathbb{E}\left[Y_{ij}^{(k)}|\boldsymbol{x}_{ij}^{(k)}\right]\right) &= \boldsymbol{x}_{ij}^{(k)}\boldsymbol{\beta} \\ \text{Var}\left[Y_{ij}^{(k)}|\boldsymbol{x}_{ij}^{(k)}\right] &= \varphi\mathcal{V}\left(\mathbb{E}\left[Y_{ij}^{(k)}|\boldsymbol{x}_{ij}^{(k)}\right]\right), \end{aligned}$$

where $\boldsymbol{x}_{ij}^{(k)}$ contains covariates associated to the j^{th} development year of the claim k. The prediction of the amount paid in the j^{th} development year for claim k having occurrence in year i is obtained with

$$\widehat{Y}_{ij}^{(k)} = g^{-1}\left(\boldsymbol{x}_{ij}^{(k)}\widehat{\boldsymbol{\beta}}\right).$$

An estimation of the reserve can be obtained by summing all individual reserves for future development years, given by Equation 1.0.4. Finally, a predictive distribution for the total reserve can be computed using simulations.

CHAPTER III

STATISTICAL LEARNING AND GRADIENT BOOSTING

The vast majority of the literature on individual loss reserving is about parametric models, which means they assume a fixed and parametric structural form. One of the main drawbacks of these methods is the lack of flexibility of the structure. Nowadays, statistical learning techniques are very popular in the field of data analytics and offer many non-parametric solutions to claim reserving. These methods give more freedom to the model and often outperform the accuracy of their parametric counterparts. However, only few non-parametric approaches have been developed using micro-level information. One of them is presented in [WU18], where the number of payments is modeled using regression trees in a discrete time framework. The occurrence of a claim payment is assumed to have a Bernoulli distribution, and the probability is then computed using a regression tree as well as available characteristics. The author uses regression trees in his article, but many other machine learning techniques, such as boosting and random forests, can be applied in order to compute the Bernoulli parameter. Other researchers [BR17] have also developed a non-parametric approach using a machine learning algorithm known as *Extra-Trees*, an ensemble of many unpruned regression trees, for loss reserving.

In this section, the general framework of supervised machine learning is presented

before introducing the generic gradient boosting algorithm. Then, a gradient boosting algorithm based on decision trees is described, and finally, we explain how gradient boosting can be used to compute reserves. A brief summary of regularization in the gradient boosting setting is also done.

3.1 Supervised Machine Learning

Supervised machine (or statistical) learning aims at learning a prediction function from labeled examples stored in a training dataset $\mathcal{D} = \{(y_i, \boldsymbol{x}_i)\}_{i=1}^{n}$. These examples are assumed to be representative of a larger population, and are used to generalize on new examples, namely to predict the response variable Y on unlabeled data points. Supervised machine learning is also used for statistical inference purposes, namely to assess the impact of one variable on another. In the dataset, both the response variable (or the dependent variable or the target variable) Y_i and the characteristics (or predictors, or features, or covariates, or independent variables) $\boldsymbol{x}_i = (x_{i1} \dots x_{ip})$ are observed by the analyst. Moreover, they can either be quantitative, categorical or ordered categorical variables. When the response is categorical, we face a classification problem, as when it is quantitative, we face a regression problem.

Models assume that the relation between the response variable and the predictors can be captured by a function f. The main objective is to obtain an approximation $\widehat{f}(\boldsymbol{x})$ of the unknown data generating function $f(\boldsymbol{x})$ mapping the covariates $\boldsymbol{x}$ to the response y. In a simplified way, it is possible to distinguish two types of models: parametric and non-parametric. In parametric models, a simple functional form for the function f is assumed, and then parameters of the model are estimated. Linear regression and generalized linear models (GLM) are examples of parametric models. In non-parametric models, no structure for the function f

is predetermined: the algorithm learns, with very few constraints, the function f. Neural networks, tree-based models and k-nearest neighbors are examples of non-parametric models. Both types have their advantages and drawbacks. Parametric models are good for interpretation due to the simple form of the link between predictors and response. Moreover, they are well suited for prediction problems for which the general form of the data generation process is known. Nevertheless, they are often less accurate than their non-parametric counterpart when the data generation function is unknown or have a complicated form. This is because non-parametric algorithms offer more flexibility and can therefore replicate a larger range of functions. On the other hand, the large number of parameters required to make the estimated function $\hat{f}$ flexible enough often makes these models too complicated to be understood.

Example 3.1.1 *In a linear regression problem, the relation is given by (we drop the reference to the subject i)*

$$Y = f(\boldsymbol{x}) + \epsilon = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \epsilon,$$

where ϵ is a random noise term, independent of $\boldsymbol{x}$, with $\mathbb{E}[\epsilon|\boldsymbol{x}] = 0$. Here, the unknown function f is assumed to belong to the class of linear functions and is characterized by its parameters $\beta_0, \ldots, \beta_p$. Since $\mathbb{E}[\epsilon|\boldsymbol{x}] = 0$, the model prediction for Y is

$$\widehat{Y} = \hat{f}(\boldsymbol{x}) = \widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \ldots + \widehat{\beta}_p x_p,$$

where $\hat{f}$ is the estimation of the function f.

Supervised machine learning algorithms form a collection of models used to estimate the function f using the data $\mathcal{D}$. Generally, a model is obtained by minimizing the expected value of a specified loss function $L(y, f(\boldsymbol{x}))$, such as the absolute

error and the squared error, over the joint distribution of $(y, \boldsymbol{x})$

$$\hat{f} = \arg\min_{f} \mathbb{E}[L(y, f(\boldsymbol{x}))]. \tag{3.1.1}$$

Since we only have access to a finite training set $\mathcal{D}$, the estimation $\hat{f}$ is obtained by minimizing the average loss function over all observations of the training set, called *empirical training risk*

$$\hat{f} = \arg\min_{f} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i)) \tag{3.1.2}$$

$$= \arg\min_{f} \sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i)). \tag{3.1.3}$$

However, for prediction purpose, we are generally not interested by the performance of the model on the training set. We rather want the model to perform well on unseen data. Overfitting happens when the model is too complex and fits the training set too well. The empirical risk on unseen data therefore starts to increase. To avoid overfitting, empirical risk is usually computed on a validation set disjoint from the training set or by using cross-validation.

Models are most of the time not perfect and make errors measured by the loss function L. The error made by a model can be broken down into two parts, namely the *reducible error* and the *irreducible error*, or the inherent uncertainty. Most of the time, $\hat{f}$ is not a perfect estimate for f, which introduces some error in the model. This is called the reducible error, since a better model could reduce this error by computing a better estimate of f. However, a model that could compute a perfect estimate of f would still make some error due to the random nature of the response variable. This is called the irreducible error, since no matter how well the model estimates f, it is impossible to reduce it. It also contains the impact of unmeasured or unmeasurable variables that are not included in $\boldsymbol{x}$.

In Example 3.1.1, if we assume a quadratic loss function $L(y, f(\boldsymbol{x})) = (f(\boldsymbol{x}) - y)^2$, we can show that the expected value of the squared error can be broken down into

the reducible and irreducible errors:

$$\begin{aligned}\mathbb{E}\left[\left(\hat{f}(\boldsymbol{x}) - Y\right)^2\right] &= \mathbb{E}\left[\left(\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x}) - \epsilon\right)^2\right] \\ &= \left(\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2 - 2\mathbb{E}[\epsilon]\left(\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\right) + \mathbb{E}\left[\epsilon^2\right] \\ &= \underbrace{\left(\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2}_{\text{Reducible error}} + \underbrace{\text{Var}[\epsilon]}_{\text{Irreducible error}}.\end{aligned}$$

3.2 Gradient Boosting

In this paper, we focus on a specific class of machine learning models called *gradient boosting* algorithms. Gradient boosting is a machine learning technique which combines sequentially many weak prediction models called *weak learners*, or *base learners*, to form a strong predictor by optimizing an objective function. In the binary classification framework, weak learners are basic classification models that are just a little better than throwing a coin, as simple classification trees. In the case of regression, weak learners are typically simple regression trees but they could be any simple regression model. Note that the gradient boosting algorithm would still work if we were using complex models as weak learners. However, this often leads to poor performance.

Recall that the aim of a machine learning technique is to obtain an approximation $\hat{f}(\boldsymbol{x})$ of the unknown data generating function $f(\boldsymbol{x})$. The gradient boosting method attempts to approximate $f(\boldsymbol{x})$ with a weighted sum of weak learners $h(\boldsymbol{x}; \boldsymbol{\theta})$

$$f(\boldsymbol{x}) = \sum_{i=1}^{M} \beta_m h\left(\boldsymbol{x}, \boldsymbol{\theta}_m\right), \tag{3.2.1}$$

where $\boldsymbol{\theta}_m$ is the set of parameters characterizing the function h. Weak learners $h(\boldsymbol{x}; \boldsymbol{\theta}_m)$ all belong to the same class of functions $\mathcal{H}$. Consequently, the minimiza-

tion problem described by Equation 3.1.3 becomes

$$\{\beta_m, \boldsymbol{\theta}_m\}_{m=1}^{M} = \underset{\{\beta'_m, \boldsymbol{\theta}'_m\}_{m=1}^{M}}{\arg\min} \sum_{i=1}^{n} L\left(y_i, \sum_{m=1}^{M} \beta'_m h(\boldsymbol{x}_i; \boldsymbol{\theta}'_m)\right). \tag{3.2.2}$$

However, this is a tremendous optimization problem and in most of the cases, this is infeasible computationally, as the next example shows.

Example 3.2.1 *We consider a dataset of size n, a quadratic loss function, a weak learner given by*

$$h(\boldsymbol{x}; \boldsymbol{\theta}_m) = \exp(\theta_{m1} x_1 + \theta_{m2} x_2)$$

and $M = 2$. The global optimization problem defined by Equation 3.2.2 becomes

$$\begin{aligned}\{\beta_m, \boldsymbol{\theta}_m\}_{m=1}^{2} = \underset{\{\beta'_m, \boldsymbol{\theta}'_m\}_{m=1}^{2}}{\arg\min} \sum_{i=1}^{n} &\left(y_i - \beta'_1 \left(\exp\left(\theta'_{11} x_{i1} + \theta'_{12} x_{i2}\right)\right)\right. \\ &\left. - \beta'_2 \left(\exp\left(\theta'_{21} x_{i1} + \theta'_{22} x_{i2}\right)\right)\right)^2.\end{aligned}$$

Even in this simplified situation, the global solution $(\beta_1, \beta_2, \theta_{11}, \theta_{12}, \theta_{21}, \theta_{22})$ is quite complex to obtain.

In theses situations, one can try a *greedy-stagewise* approach called *forward stagewise modeling* in solving, for $m = 1, \ldots, M$,

$$(\beta_m, \boldsymbol{\theta}_m) = \underset{\beta, \boldsymbol{\theta}}{\arg\min} \sum_{i=1}^{n} L\left(y_i, f_{m-1}(\boldsymbol{x}_i) + \beta h(\boldsymbol{x}_i; \boldsymbol{\theta})\right), \tag{3.2.3}$$

and by updating the model with

$$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \beta_m h(\boldsymbol{x}; \boldsymbol{\theta}_m), \tag{3.2.4}$$

starting with an initial value $f_0(\boldsymbol{x})$. This approach is *greedy* because at each step, it finds the optimal local solution without worrying about the next steps. *Stagewise* means that the model is constructed step by step by adding a new function at each iteration without modifying previous functions.

Example 3.2.2 *In Example 3.2.1, the optimization problem becomes a local procedure:*

$$\begin{aligned}(\beta_1, \boldsymbol{\theta}_1) &= \arg\min_{\beta', \boldsymbol{\theta}'} \sum_{i=1}^{n} L(y_i, f_0(\boldsymbol{x}_i) + \beta' h(\boldsymbol{x}_i; \boldsymbol{\theta}')) \\ &= \arg\min_{\beta', \boldsymbol{\theta}'} \sum_{i=1}^{n} \left(y_i - f_0(\boldsymbol{x}_i) - \beta' \left(\exp\left(\theta_1' x_{i1} + \theta_2' x_{i2}\right)\right)\right)^2\end{aligned}$$

and

$$\begin{aligned}(\beta_2, \boldsymbol{\theta}_2) &= \arg\min_{\beta', \boldsymbol{\theta}'} \sum_{i=1}^{n} L(y_i, f_0(\boldsymbol{x}_i) + \beta_1 h(\boldsymbol{x}_i; \boldsymbol{\theta}_1) + \beta' h(\boldsymbol{x}_i; \boldsymbol{\theta}')) \\ &= \arg\min_{\beta', \boldsymbol{\theta}'} \sum_{i=1}^{n} \left(y_i - f_0(\boldsymbol{x}_i) - \beta_1 \left(\exp\left(\theta_{11} x_{i1} + \theta_{12} x_{i2}\right)\right)\right. \\ &\quad \left. - \beta' \left(\exp\left(\theta_1' x_{i1} + \theta_2' x_{i2}\right)\right)\right)^2,\end{aligned}$$

which is easier to compute.

For some choices of loss functions and/or weak learners, the solution to Equation 3.2.3 can be hard to obtain. In those cases, we need another method in order to find optimal parameters.

Based on the steepest-descent minimization method, [FR01] suggests replacing Equation 3.2.3 by a two-step approach:

1. Evaluate the negative gradient of the loss function based on the data

$$-\boldsymbol{g}_m = \{-g_m(\boldsymbol{x}_i)\}_{i=1}^n,$$

where

$$-g_m(\boldsymbol{x}_i) = -\left[\frac{\partial L(y_i, f(\boldsymbol{x}_i))}{\partial f(\boldsymbol{x}_i)}\right]_{f(\boldsymbol{x}_i)=f_{m-1}(\boldsymbol{x}_i)}.$$

It gives the step direction of the steepest descent at the point $f(\boldsymbol{x}_i) = f_{m-1}(\boldsymbol{x}_i)$ of the n-dimensional data space. This negative gradient is unconstrained since no particular structure is assumed for the function f. However, this gradient is only defined at the data points $\{\boldsymbol{x}_i\}_{i=1}^n$, which means it can not be generalized to other data points than those of the training set. The solution proposed by [FR01] is to approximate the unconstrained negative gradient by a weak learner h choosen as the closest one to the negative gradient in the L^2 sense. It is called the constrained negative gradient and can be obtained by solving

$$\boldsymbol{\theta}_m = \arg\min_{\boldsymbol{\theta},\beta} \sum_{i=1}^{n} (-g_m(\boldsymbol{x}_i) - \beta h(\boldsymbol{x}_i;\boldsymbol{\theta}))^2, \tag{3.2.5}$$

which is equivalent to fit a weak learner h by least square on the training set with responses $\{-g_m(\boldsymbol{x}_i)\}_{i=1}^n$, called *pseudo residuals.*

2. The best step size

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{n} L(y_i, f_{m-1}(\boldsymbol{x}_i) + \rho h(\boldsymbol{x}_i;\boldsymbol{\theta}_m)) \tag{3.2.6}$$

is computed and the prediction function is updated:

$$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \rho_m h(\boldsymbol{x};\boldsymbol{\theta}_m). \tag{3.2.7}$$

In some rare cases as in the AdaBoost algorithm (see [FS97]), the optimal step size has a closed form. However, optimal or sub-optimal step size ρ_m is most of the time found using line search [ST03]. This leads to Algorithm 1, compatible with any differentiable loss function $L(y, f(\boldsymbol{x}))$ and any weak learner $h(\boldsymbol{x};\boldsymbol{\theta})$.

Example 3.2.3 *Squared-error loss $L(y, f(\boldsymbol{x})) = \frac{1}{2}(y - f(\boldsymbol{x}))^2$ is a popular choice of loss function for regression. When this particular loss is used, Algorithm 1*

Algorithm 1 *Generic Gradient Boosting*

1. Initialize $f_0(\boldsymbol{x}) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$.

2. For $m = 1, \ldots, M$, do:

   (a) compute pseudo residuals

   $$-g_m(\boldsymbol{x}_i) = -\left[\frac{\partial L(y_i, f(\boldsymbol{x}_i))}{\partial f(\boldsymbol{x}_i)}\right]_{f(\boldsymbol{x})=f_{m-1}(\boldsymbol{x})}, \; i = 1, \ldots, n;$$

   (b) find the parameters of the weak learner that best fits the pseudo residuals

   $$\boldsymbol{\theta}_m = \arg\min_{\boldsymbol{\theta},\beta} \sum_{i=1}^{n} (-g_m(\boldsymbol{x}_i) - \beta h(\boldsymbol{x}_i))^2;$$

   (c) find the optimal step size

   $$\rho_m = \arg\min_{\rho} \sum_{i=1}^{n} L(y_i, f_{m-1}(\boldsymbol{x}_i) + \rho h(\boldsymbol{x}_i; \boldsymbol{\theta}_m));$$

   (d) update prediction function

   $$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \rho_m h(\boldsymbol{x}; \boldsymbol{\theta}_m).$$

3. Final prediction function is $f_M(\boldsymbol{x})$.

simplifies a bit. First, we notice that the pseudo residuals $\{-g_m(\boldsymbol{x}_i)\}_{i=1}^n$ *become simply the residuals* $\{y_i - f_{m-1}(\boldsymbol{x}_i)\}_{i=1}^n$. *Indeed,*

$$\begin{aligned} -g_m(\boldsymbol{x}_i) &= -\left[\frac{\partial L(y_i, f(\boldsymbol{x}_i)}{\partial f(\boldsymbol{x}_i)}\right]_{f(\boldsymbol{x})=f_{m-1}(\boldsymbol{x})} \\ &= -\left[\frac{\partial \frac{1}{2}(y_i - f(\boldsymbol{x}_i))^2}{\partial f(\boldsymbol{x}_i)}\right]_{f(\boldsymbol{x})=f_{m-1}(\boldsymbol{x})} \\ &= [y_i - f(\boldsymbol{x}_i)]_{f(\boldsymbol{x})=f_{m-1}(\boldsymbol{x})} \\ &= y_i - f_{m-1}(\boldsymbol{x}_i). \end{aligned}$$

Also, the minimization in Equation 3.2.6 produces the results $\rho_m = \beta_m$, *where* β_m *is the beta minimizing the expression in Equation 3.2.5. The initial guess for the prediction function* $f_0(\boldsymbol{x})$ *becomes the mean of response variables over the* n *observations. Therefore, with squared-error loss, gradient boosting becomes a stagewise approach that fits iteratively the residuals of the previous model using a weak learner. Least square gradient boosting is described in Algorithm 2.*

3.2.1 TreeBoost

A decision tree partitions the predictor space $\mathcal{X}$ into J subspaces $\{R_j\}_{j=1}^J$ called *regions.* In the regression context, a prediction constant b_j is assigned to each region, and the prediction function has an additive form given by

$$h(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{j=1}^{J} b_j \mathbb{1}(\boldsymbol{x} \in R_j),$$

where $\boldsymbol{\theta} = \{R_j, b_j\}_{j=1}^J$ parametrizes the tree. Trees are usually fit using a top-down greedy approach called CART methodology, described by [BR84]. Gradient boosting models are usually fit using decision trees as weak learners since they show good performance. In that case, the update in Equation 3.2.4 becomes

$$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \rho_m \sum_{j=1}^{J_m} b_{jm} \mathbb{1}(\boldsymbol{x} \in R_{jm}), \tag{3.2.8}$$

Algorithm 2 *Least Square Gradient Boosting*

1. Initialize $f_0(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} y_i$.

2. For $m = 1, \ldots, M$, do:

   (a) compute residuals

   $$r_i = y_i - f_{m-1}(\boldsymbol{x}_i),\ i = 1, \ldots, n;$$

   (b) find parameters of the weak learner and step size that best fit the residuals

   $$(\boldsymbol{\theta}_m, \rho_m) = \arg\min_{\boldsymbol{\theta},\rho} \sum_{i=1}^{n} (r_i - \rho h(\boldsymbol{x}_i; \boldsymbol{\theta}))^2;$$

   (c) update prediction function

   $$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \rho_m h(\boldsymbol{x}; \boldsymbol{\theta}_m).$$

3. Final prediction function is $f_M(\boldsymbol{x})$.

where $\{R_{jm}\}_{j=1}^{J_m}$ are the regions created by the m^{th} tree to predict pseudo residuals $\{-g_m(\boldsymbol{x}_i)\}_{i=1}^{n}$ by least-squares. $\{b_{jm}\}_{j=1}^{J_m}$ are the coefficients minimizing the squared prediction error in each of the regions. At the m^{th} iteration, b_{jm} is therefore the average pseudo residual value for all observations belonging to the region R_{jm},

$$b_{jm} = \frac{\sum_{\mathcal{S}_{jm}} - g_m(\boldsymbol{x}_i)}{|\mathcal{S}_{jm}|}, \; j = 1, \ldots, J_m, \tag{3.2.9}$$

where $\mathcal{S}_{jm} = \{i : \boldsymbol{x}_i \in R_{jm}\}$ and $|\mathcal{S}_{jm}|$ are respectively the set of observations belonging to the region R_{jm} and the number of observations in region R_{jm}. The scaling factor ρ_m is obtained using Equation 3.2.6. For the special case of trees as weak learners, [FR01] proposed a modified version of the gradient boosting algorithm, called *TreeBoost*. In fact, it is also possible in Equation 3.2.8 to enter the term ρ_m into the sum, to set $\gamma_{jm} = \rho_m b_{jm}$ and to write, in an alternative way,

$$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{1}(\boldsymbol{x} \in R_{jm}). \tag{3.2.10}$$

Instead of adding only one weak learner, Equation 3.2.8 can now be seen as adding J_m separate weak learners at each iteration. The optimal coefficients $\{\gamma_{jm}\}_{j=1}^{J_m}$ are the solution to

$$\{\gamma_{jm}\}_{j=1}^{J_m} = \underset{\{\gamma_j\}_{j=1}^{J_m}}{\arg\min} \sum_{i=1}^{n} L\left(y_i, f_{m-1}(\boldsymbol{x}_i) + \sum_{j=1}^{J_m} \gamma_j \mathbb{1}(\boldsymbol{x} \in R_{jm})\right). \tag{3.2.11}$$

Since the regions produced by decision trees are disjoints, optimal coefficients can be found separately, with

$$\gamma_{jm} = \underset{\gamma}{\arg\min} \sum_{\mathcal{S}_{jm}} L\left(y_i, f_{m-1}(\boldsymbol{x}_i) + \gamma\right), \tag{3.2.12}$$

which is the optimal constant value for each leaf of the tree given the prediction function $f_{m-1}(\boldsymbol{x})$. This modified version of the gradient boosting algorithm

estimates the optimal coefficient of each separate region of the tree instead of estimating one coefficient for the whole tree, which improves the quality of the fit. Number of regions made by the tree at each iteration is often fixed, so $J_m = J$, for $m = 1, \ldots, M$. *TreeBoost* algorithm is detailed in Algorithm 3. It should be noted that the number of trees M as well as the number of regions in each tree J are treated as hyperparameters, and their optimal values are most of the time estimated using cross-validation.

Algorithm 3 *TreeBoost*

1. Initialize $f_0(\boldsymbol{x}) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$.

2. For $m = 1, \ldots, M$, do:

   (a) compute pseudo residuals

   $$-g_m(\boldsymbol{x}_i) = -\left[\frac{\partial L(y_i, f(\boldsymbol{x}_i))}{\partial f(\boldsymbol{x}_i)}\right]_{f(\boldsymbol{x})=f_{m-1}(\boldsymbol{x})}, \; i = 1, \ldots, n;$$

   (b) fit a tree to the data $\{(\boldsymbol{x}_i, -g_m(\boldsymbol{x}_i))\}_{i=1}^{n}$, which gives regions $\{R_{jm}\}_{j=1}^{J}$;

   (c) compute optimal coefficient for each region

   $$\gamma_{jm} = \arg\min_{\gamma} \sum_{\boldsymbol{S}_{jm}} L\left(y_i, f_{m-1}(\boldsymbol{x}_i) + \gamma\right), \; j = 1, \ldots, J;$$

   (d) update prediction function

   $$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \sum_{j=1}^{J} \gamma_{jm} \mathbb{1}(\boldsymbol{x} \in R_{jm}).$$

3. Final prediction function is $f_M(\boldsymbol{x})$.

3.2.2 Regularization

Regularization refers to methods used to prevent overfitting by constraining the fitting procedure, and is a key idea in prediction models. Each new weak learner added to the gradient boosting model reduces the average loss function over the training data, which is the empirical training risk. Consequently, if M is chosen large enough, it is possible to make the training risk arbitrarily small. However, fitting the training data too closely degrades the model's generalization ability and increases the risk on unseen data points. A way to regularize is therefore to control the value of M. The goal is to find the value of M that minimizes the risk on future predictions, and a way of doing this is by cross-validation [FH01].

Another way to regularize is to slow the rate at which the algorithm is learning from the training data at each iteration, called *shrinkage*. This is done by introducing a shrinkage parameter ν, more commonly called *learning rate*, and by replacing the update in Equation 3.2.4 by

$$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \nu\beta_m h(\boldsymbol{x}; \boldsymbol{\theta}_m), \quad 0 < \nu \leq 1. \tag{3.2.13}$$

Therefore, at each iteration, the new weak learner added is simply scaled by the learning rate. The smaller the learning rate, the slower the algorithm learns. Thus, decreasing the value of the parameter ν increases the optimal value for M, which means these two parameters must be optimized jointly, for instance with cross-validation. It has been found that shrinkage improves dramatically the performance of gradient boosting, and yields better results than only restricting the number of weak learners (see [CO83]). Many other regularization methods exist for gradient boosting, but shrinkage is certainly the one that leads to the best improvement.

3.3 Gradient Boosting for Loss Reserving

In order to train gradient boosting models, we use an algorithm called *XGBoost* developed by [CG16], and regression trees are chosen as weak learners. Moreover, loss function used is the squared loss $L(y, f(x)) = (y - f(x))^2$ but other options such as residual deviance for gamma regression were considered without significantly altering the conclusions. We postpone to a subsequent case study a more detailed analysis of the impact of the choice of this function. Models are implemented in `R` thanks to `caret` package. *XGBoost* is similar to *TreeBoost* algorithm described in Section 3.2.1 and thus follows the principles of boosting. The differences between the two algorithms are mostly in modeling details. Also, *XGBoost* usually yields more accurate predictions, requires less computational resources and is faster to fit. For more details about *XGBoost*, see [CG16].

Let us say we have a portfolio of claimants $\mathcal{S}$ on which we want to train an *XGBoost* model for loss reserving. In order to predict total paid amount for a claim k, we use information we have about the case at evaluation date t^*, denoted by $\boldsymbol{x}_{t^*}^{(k)}$. The *XGBoost* algorithm therefore learns a prediction function $\hat{f}_{XGB}$ on the dataset $\mathcal{D}_{t^*}^{(\mathcal{S})} = \{(\boldsymbol{x}_{t^*}^{(k)}, C_{T_3}^{(k)})\}_{k \in \mathcal{S}}$. Then, the predicted total paid amount for claim k is given by

$$\widehat{C}_{T_3}^{(k)} = \hat{f}_{XGB}\left(\boldsymbol{x}_{t^*}^{(k)}\right).$$

Reserve for claim k is

$$\widehat{R}_{t^*}^{(k)} = \widehat{C}_{T_3}^{(k)} - C_{t^*}^{(k)},$$

and the RBNS reserve for the whole portfolio is computed with

$$\widehat{R}_{t^*} = \sum_{k \in \mathcal{S}} \widehat{R}_{t^*}^{(k)}.$$

Gradient boosting is a non-parametric algorithm and no distribution is assumed for the response variable. Therefore, in order to compute the variance of the reserve and some risk measures, we use a non-parametric bootstrap procedure.

CHAPTER IV

ANALYSIS

In this chapter, we present an extended case study based on a detailed dataset from a property and casualty insurance company. In Section 4.1, we describe the dataset; in Section 4.2, we present the covariates used in the models; in Section 4.3, we explain how we construct our models and how we train them. Finally, in Section 4.4, we present our numerical results and our analyzes.

4.1 Data

We have at our disposal a database consisting of 67,203 Accident Benefit claims involving 82,520 claimants from a North American insurance company, running from January 1st 2004 to December 31st 2016. We therefore let τ, the *ad hoc* starting point, be January 1st 2004. The portfolio containing the 82,520 claimants is denoted $\mathcal{S}$. Most claims involve one (83%) or two (13%) insured (see Figure 4.1.1), and the maximum observed number of claimants for a claim is 9. Consequently, there is a possibility of dependence between some payments in the database. Nevertheless, we assume in this paper that all claimants are independent and we postpone the analysis of this dependence. The mean incurred as of December 31st 2016 is \$16,067, and the incurred distribution for each occurrence year is presented in Figure 4.1.2. About 40% of the claims close with a total paid amount of zero. If

we exclude claims with an incurred of zero as of December 31st 2016, the average incurred increases to \$24,144.

The data is longitudinal, and each row of the database corresponds to a snapshot of a file. For each element in $\mathcal{S}$, a snapshot is taken at the end of every quarter, and we have information from the reporting date until December 31st 2016. Therefore, a claimant is represented by a maximum of 52 rows, since 52 is the number of quarters between January 1st 2004 and December 31st 2016. A line is added in the database even if there is no new information, i.e., it could be possible that two consecutive lines provide precisely the same information. This method of storing claim data is not unique and varies from insurer to insurer. In Chapter 1, we have seen that intrinsically, the loss reserving problem is in continuous time. However, since we only have discrete information, we will now consider it a discrete problem, which means the constructed models will be discrete.

Example 4.1.1 *All claimants with reporting date on the first quarter of year 2004 are represented by 52 rows, those with reporting date on the second quarter of year 2004 are represented by 51 rows, etc. Finally, all claimants with reporting date on the last quarter of year 2016 are represented by only one row.*

The information vector for claim k at time t is given by $\mathcal{D}_t^{(k)} = (\boldsymbol{x}_t^{(k)}, C_t^{(k)})$. Therefore, information matrix about the whole portfolio at time t is given by $\mathcal{D}_t^{(\mathcal{S})} = \{\mathcal{D}_t^{(k)}\}_{k\in\mathcal{S}}$. Because the database contains a snapshot for each claimant at each quarter, it contains information $\mathcal{D}_{\mathcal{S}} = \{\mathcal{D}_t^{(\mathcal{S})}\}_{\{0.25t:\, t\in\mathbb{N},\, t\leq 52\}}$, where t is the number of years since τ. Claimant features include characteristics of the claimant himself, but also those regarding the claim and the policy associated with him. In order to validate the models, we need to know how much has actually been paid for each claim. In portfolio $\mathcal{S}$, total paid amount C_{T_3} is still unknown for 19% of the claimants, because they are related to claims that are

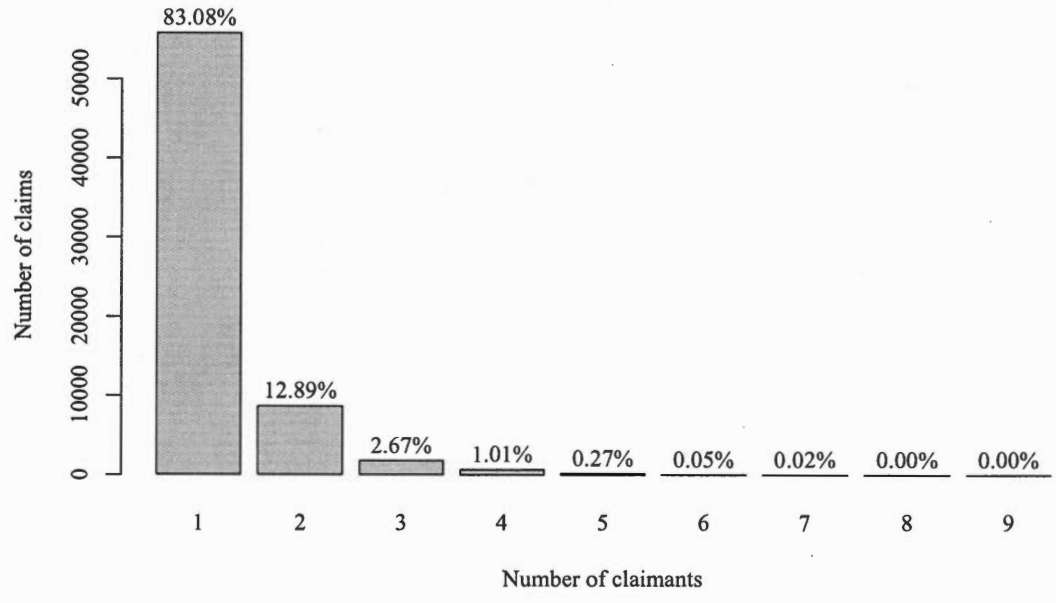

Figure 4.1.1: *Number of claimants per claim.*

open on December 31st 2016 (see Figure 4.1.3). In Figure 4.1.3, we can see that open claims are mostly from recent accident years. To overcome this issue, we use a subset $\mathcal{S}_7 = \{k \in \mathcal{S} : T_1^{(k)} < 7\}$ of $\mathcal{S}$, i.e., we consider only accident years from 2004 to 2010 for both training and validation. This subset contains 36,843 claimants related to 30,483 claims. That way, only 0.67% of the claimants are associated with claims that are still open as of the end of 2016, so we know the exact total paid amount for 99.33% of them, assuming no reopening after 2016. For the small proportion of open claims, we assume that the incurred amount set by experts is the true total paid amount. Hence, the evaluation date is fixed to December 31st 2010 and we set $t^* = 7$. This is the date at which the RBNS reserve must be evaluated for claimants in $\mathcal{S}_7$. This implies that the models are not allowed to use information past this date for their training. Information after the evaluation date is only used for validation. A summary about sets $\mathcal{S}$ and $\mathcal{S}_7$

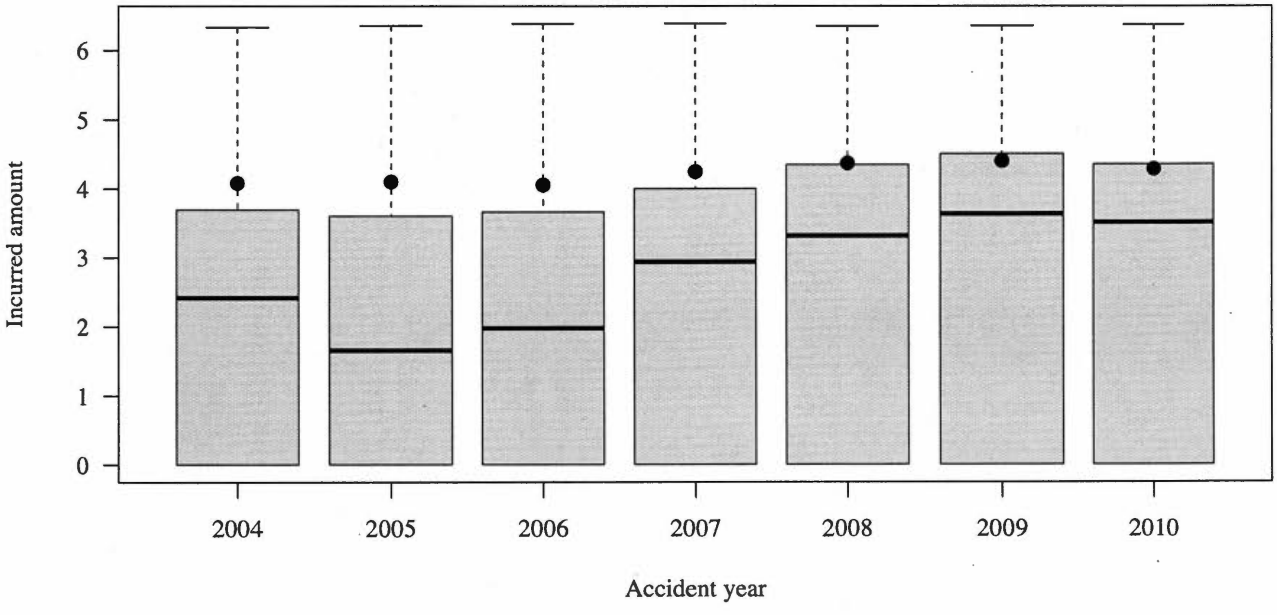

Figure 4.1.2: *Distribution of final incurred by accident years (on a base 10 log scale). The first quartile is equal to the minimum for all accident years since many claims close at zero. The average incurred for each accident year is represented by a dot.*

is done in Table 4.1.1.

For simplicity and for computational purpose, the quarterly database is summarized to form a yearly database $\mathcal{D}_{\mathcal{S}_7} = \{\mathcal{D}_t^{(\mathcal{S}_7)}\}_{t=1}^{13}$, where $\mathcal{D}_t^{(\mathcal{S}_7)} = \{\mathcal{D}_t^{(k)}\}_{k\in\mathcal{S}_7}$. 70% of the 36,843 claimants have been sampled randomly to form the training set of indices $\mathcal{T} \subset \mathcal{S}_7$, and the other 30% forms the validation set of indices $\mathcal{V} \subset \mathcal{S}_7$, which gives the training and validation datasets $\mathcal{D}_\mathcal{T} = \{\mathcal{D}_t^{(\mathcal{T})}\}_{t=1}^{13}$ and $\mathcal{D}_\mathcal{V} = \{\mathcal{D}_t^{(\mathcal{V})}\}_{t=1}^{13}$. A summary of training and validation datasets is done in Table 4.1.2.

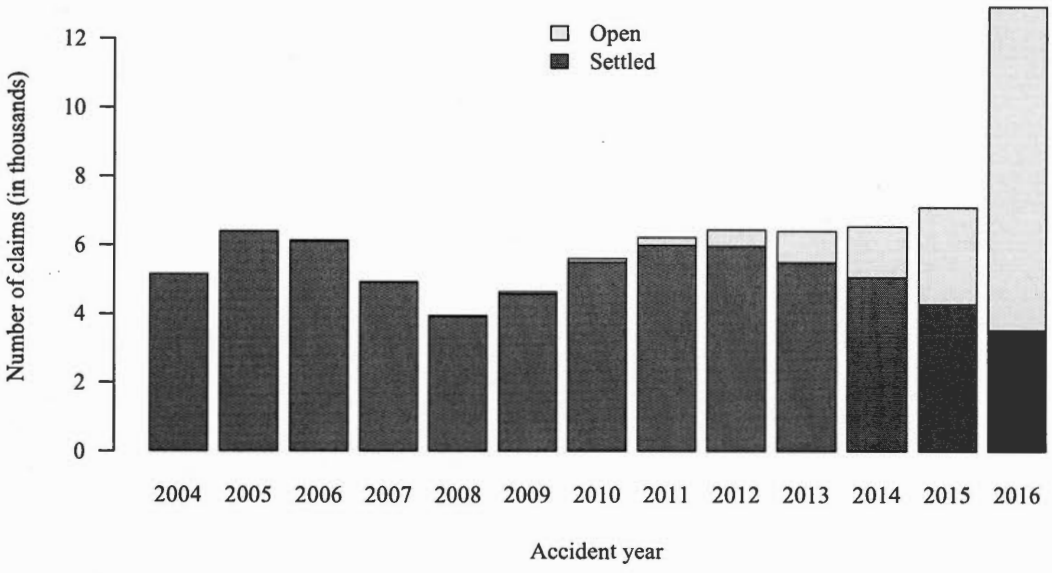

Figure 4.1.3: *Status of claims on December 31st 2016.*

4.2 Covariates

In partnership with the insurance company, we have selected 20 covariates in order to predict total paid amount for each of the claimants, described in Table 4.2.1. Note that some variables have been censored due to the confidentiality agreement. To make models comparable, we use the same covariates for all of them. Some covariates are characteristics of the claimant and some are about the claim itself, as the accident year and the development year. Some of the covariates, such as the accident year, are static, which means their value do not change over time. These covariates are quite easy to handle because their final value is known since the reporting of the accident. However, some part of available information is expected to develop between t^* and the closure date. More precisely, 8 of the 20 covariates are dynamic variables, as the number of healthcare providers. To handle those

Table 4.1.1: *Comparison of complete set of claimants ($\mathcal{S}$) and set of claimants from accident years 2004-2010 ($\mathcal{S}_7$). % of open claims is on December 31st 2016.*

Dataset	# claims	# claimants	% of open claims
$\mathcal{S}$	67,203	82,520	18.87
$\mathcal{S}_7$	30,483	36,843	0.67

Table 4.1.2: *Details about training and validation datasets.*

Dataset	# claims	# claimants	# rows
$\mathcal{D}_\mathcal{T}$	22,096	25,790	256,894
$\mathcal{D}_\mathcal{V}$	10,164	11,053	109,918

dynamic covariates, we have, at least, the following two options:

- we can assume that they are static, which can lead to a bias in the predictions obtained, or;
- we can, for each of these variables, (1) adjust a dynamic model, (2) obtain a prediction of the complete trajectory, and (3) use the algorithm conditionally to the realization of this trajectory. Moreover, it is possible that there is some dependence between these variables and therefore a multivariate approach could be necessary.

In this work, we simply assume that values of dynamic covariates are frozen at time t^*. Also, notice that the case reserve set by adjusters is not used as a covariate. This information would probably be useful, but would make models non self-sufficient.

Table 4.2.1: *Covariates used in the models.*

Description	Type
Accident year	static
Development year	dynamic
Indicator of the status "settled" of the claim	dynamic
Number of wounded	static
Number of healthcare providers	dynamic
Covariate 6	static
Covariate 7	static
Covariate 8	static
Covariate 9	static
Covariate 10	dynamic
Covariate 11	static
Covariate 12	static
Covariate 13	static
Covariate 14	static
Covariate 15	static
Covariate 16	static
Covariate 17	dynamic
Covariate 18	dynamic
Covariate 19	dynamic
Covariate 20	dynamic

4.3 Training of *XGBoost* models

In Section 3.3, we have seen that in order to fit an *XGBoost* model, we need the training response C_{T_3} for each claimant in the training set. However, 22% of the claimants in $\mathcal{S}_7$, mostly from recent accident years, are associated with claims that are still not settled at $t^* = 7$, and their total paid amount C_{T_3} is still unknown (see Figure 4.3.1). We therefore face a censored response variable issue. At this stage, several options are available:

1. The simplest solution is to train the model on a training set where only settled claims (or non censored claims) are included. Hence, the response

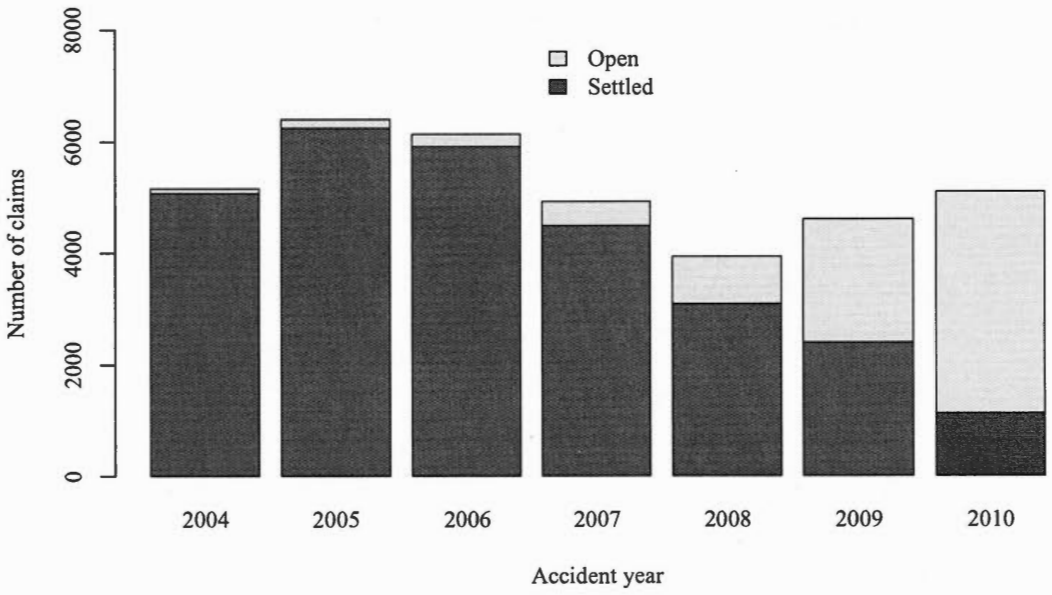

Figure 4.3.1: *Status of claims on December* 31^{st} *2010.*

is known for all claimants. However, this leads to a selection bias because claims that are already settled at t^* tend to have shorter developments, and claims with shorter developments tend to have lower total paid amounts. Consequently, the model is almost exclusively trained on simple claims with low training responses, and this leads to underestimation of the total paid amount for new claims. Furthermore, a significant proportion of the claimants are removed from the analysis, which causes a loss of information. We will further analyze this bias below.

2. In [LM16], a different and interesting approach is proposed: in order to correct the selection bias induced by the presence of censored data, a strategy called "Inverse Probability of Censoring Weighting" (IPCW) is implemented, which involves assigning weights to observations to offset the lack of com-

plete observations in the sample. The weights are determined using the Kaplan-Meier estimator of the censoring distribution and a modified CART algorithm is used to make the predictions. We refer an interested reader to the aforementioned paper.

3. A third approach is to develop open claims at t^* using parameters from a classical approach such as the Mack's model and GLM. We also discuss in more detail this idea below.

In order to train an *XGBoost* model, we have at our disposition the training dataset $\mathcal{D}_{\mathcal{T}} = \{(\boldsymbol{x}_t, C_t)\}_{t=1}^{13}$. All models are adjusted using the same 20 covariates described in section 4.2. Because some covariates are dynamic, the design matrix $\boldsymbol{x}_t$ changes over time, that is to say $\boldsymbol{x}_t \neq \boldsymbol{x}_{t'}$ for $t \neq t'$. Unless otherwise stated, the models are all trained using $\boldsymbol{x}_7$, which is the latest information we have about claimants assuming information after $t^* = 7$ is unknown.

In practice, the evaluation date is always in the present, which means that the total paid amount is still unknown for many claims. In that case, a model using real responses is not applicable. In this work, we fix the evaluation date in the past in order to know the total paid amount for each claim, which makes possible the training of such a model. This **Model A** acts as a benchmark model in our case study because it is fit using C_{13} as training responses and it is the best model we can hope for. Therefore, in order to train **Model A**, data $\mathcal{D}_{\mathcal{T}}^A = \{(\boldsymbol{x}_7^{(k)}, C_{13}^{(k)})\}_{k \in \mathcal{T}}$ is inputted into the *XGBoost* algorithm, that learns the prediction function $\widehat{f}_A$.

Model B, which is a biased one, is fit using C_7 as training responses, but only on the set of claimants for which the claim is settled at time $t^* = 7$. Hence, **Model B** is trained using $\mathcal{D}_{\mathcal{T}}^B = \{(\boldsymbol{x}_7^{(k)}, C_7^{(k)})\}_{k \in \mathcal{T}_B}$, where $\mathcal{T}_B = \{k \in \mathcal{T} : T_3^{(k)} < 7\}$, giving the prediction function $\widehat{f}_B$. This model allows us to measure the extent of the selection bias.

In the next models, we develop claims at t^*, i.e., we predict *pseudo-responses* $\widehat{C}_{T_3}$ using training set $\mathcal{D}_{\mathcal{T}}$, and these pseudo-responses are subsequently used to fit the model.

In **Model C1**, **Model C2** and **Model C3**, claims are developed using the chain-ladder algorithm and the Mack's model with only accident years and development years as covariates. In the first case, we use expected values as pseudo-responses while in the second case and in the third case, we use a bootstrap procedure in order to obtain more conservative pseudo-responses. More specifically, information from data $\{\mathcal{D}_t^{(\mathcal{T})}\}_{t=1}^7$ is aggregated by accident year and by development year to form the cumulative run-off triangle

$$\boldsymbol{C} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} & C_{17} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} & \cdot \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & \cdot & \cdot \\ C_{41} & C_{42} & C_{43} & C_{44} & \cdot & \cdot & \cdot \\ C_{51} & C_{52} & C_{53} & \cdot & \cdot & \cdot & \cdot \\ C_{61} & C_{62} & \cdot & \cdot & \cdot & \cdot & \cdot \\ C_{71} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix},$$

where C_{ij} is the cumulative aggregate payment for claims of accident year i at development year j. For the **Model C1**, the chain-ladder algorithm is applied on this triangle in order to obtain pseudo-responses $\{\widehat{C}_{T_3}^{(k)}\}_{k\in\mathcal{T}}$

$$\widehat{C}_{T_3}^{(k)} = \widehat{\lambda}_j^c C_7^{(k)}, \text{ where } \widehat{\lambda}_j^c = \prod_{l=j}^{6} \widehat{\lambda}_l \tag{4.3.1}$$

and

$$\widehat{\lambda}_j = \frac{\sum_{i=1}^{7-j} C_{i(j+1)}}{\sum_{i=1}^{7-j} C_{ij}}, \tag{4.3.2}$$

for $j = 1, \ldots, 6$. For closed claims, we simply set $\widehat{C}_{T_3}^{(k)} = C_7^{(k)}$. For the **Model C3** and the **Model C2**, we consider the bootstrap approach described in [EV02] and

involving Pearson's residuals to generate $B = 1000$ bootstrap samples of triangles $\{\boldsymbol{C}^{(b)}\}_{b=1}^{B}$. On each of those triangles, the chain-ladder algorithm is applied to obtain the matrix of estimated development factors

$$\left[\hat{\boldsymbol{\lambda}}_1 \;\; \hat{\boldsymbol{\lambda}}_2 \;\; \hat{\boldsymbol{\lambda}}_3 \;\; \hat{\boldsymbol{\lambda}}_4 \;\; \hat{\boldsymbol{\lambda}}_5 \;\; \hat{\boldsymbol{\lambda}}_6\right] = \begin{bmatrix} \hat{\lambda}_1^{(1)} & \hat{\lambda}_2^{(1)} & \hat{\lambda}_3^{(1)} & \hat{\lambda}_4^{(1)} & \hat{\lambda}_5^{(1)} & \hat{\lambda}_6^{(1)} \\ \hat{\lambda}_1^{(2)} & \hat{\lambda}_2^{(2)} & \hat{\lambda}_3^{(2)} & \hat{\lambda}_4^{(2)} & \hat{\lambda}_5^{(2)} & \hat{\lambda}_6^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hat{\lambda}_1^{(B)} & \hat{\lambda}_2^{(B)} & \hat{\lambda}_3^{(B)} & \hat{\lambda}_4^{(B)} & \hat{\lambda}_5^{(B)} & \hat{\lambda}_6^{(B)} \end{bmatrix}.$$

In order to have conservative pseudo-responses, we choose the development factor $\widehat{\lambda}_j$ to be the 80^{th} and 90^{th} empirical quantile of vector $\widehat{\boldsymbol{\lambda}}_j$, $j = 1, \ldots, 6$. Cumulative development factors can be computed with Equation 4.3.1 and, finally, pseudo-responses $\{\widehat{C}_{T_3}^{(k)}\}_{k \in \mathcal{T}}$ are obtained in the same way as in **Model C1**.

Model D1 uses an individual Poisson GLM with predictors to compute pseudo-responses, described in Section 2.2. GLM is fit on data $\{(\boldsymbol{x}_t^{(\mathcal{T})}, \boldsymbol{Y}_t^{(\mathcal{T})})\}_{t=1}^{7}$, where $\boldsymbol{x}_t^{(\mathcal{T})} = \{\boldsymbol{x}_t^{(k)}\}_{k \in \mathcal{T}}$, $\boldsymbol{Y}_t^{(\mathcal{T})} = \{Y_t^{(k)}\}_{k \in \mathcal{T}}$ and $Y_t^{(k)}$ is the yearly aggregate payment at year t for claimant k. Logarithm link function is used and coefficients are estimated by maximizing Poisson log-likelihood. Therefore, the estimation of the mean for a new observation is given by

$$\widehat{\mu}_t^{(k)} = \exp\left(\boldsymbol{x}_t^{(k)} \widehat{\boldsymbol{\beta}}\right),$$

and prediction is made according to the estimated mean:

$$\widehat{Y}_t^{(k)} = \widehat{\mu}_t^{(k)}.$$

Prediction is done for all claimants of the training set $\mathcal{T}$ for calendar years after the evaluation date, namely for $t = 8, \ldots, 13$, yielding the predictions $\{\widehat{Y}_t^{(k)}\}_{t=8}^{13}$, for all $k \in \mathcal{T}$. For all files, the pseudo-response is obtained by adding all yearly aggregate estimated payments after $t^* = 7$ to the amount already paid at the

censoring date. For claimant k,

$$\widehat{C}_{T_3}^{(k)} = C_7^{(k)} + \sum_{t=8}^{13} \widehat{Y}_t^{(k)}.$$

In **Model D1**, **Model D2**, **Model D3**, **Model D4** and **Model D5**, claims are projected using an individual quasi-Poisson GLM using all 20 covariates. More specifically, **Model D1** uses an individual Poisson GLM with all predictors to estimate the training dependent variable. The GLM is fit on data $\{(\boldsymbol{x}_t^{(\mathcal{T})}, \boldsymbol{Y}_t^{(\mathcal{T})})\}_{t=1}^{7}$, where $\boldsymbol{x}_t^{(\mathcal{T})} = \{\boldsymbol{x}_t^{(k)}\}_{k\in\mathcal{T}}$, $\boldsymbol{Y}_t^{(\mathcal{T})} = \{Y_t^{(k)}\}_{k\in\mathcal{T}}$ and $Y_t^{(k)}$ is the yearly aggregate payment at year t for claimant k. A logarithm link function is used and coefficients are estimated by maximizing the Poisson log-likelihood function. Therefore, the estimation of the expected value for a new observation is given by

$$\widehat{\mu}_t^{(k)} = \exp\left(\boldsymbol{x}_t^{(k)}\widehat{\boldsymbol{\beta}}\right),$$

and a prediction is made according to the estimated mean $\widehat{Y}_t^{(k)} = \widehat{\mu}_t^{(k)}$. Prediction is done for all claimants of the training set $\mathcal{T}$ for calendar years after the evaluation date, namely for $t = 8, \ldots, 13$, yielding the predictions $\{\widehat{Y}_t^{(k)}\}_{t=8}^{13}$, for all $k \in \mathcal{T}$. For all files of the dataset, the pseudo-response is obtained by adding all yearly estimated payments after $t^* = 7$ to the amount already paid at the censoring date. For claimant k, we have

$$\widehat{C}_{T_3}^{(k)} = C_7^{(k)} + \sum_{t=8}^{13} \widehat{Y}_t^{(k)}.$$

Model D2, **Model D3**, **Model D4** and **Model D5** are similar to **Model D1**, but instead of a Poisson GLM, a quasi-Poisson GLM is used. Moreover, rather than using $\widehat{Y}_t^{(k)} = \widehat{\mu}_t^{(k)}$ we choose $\widehat{Y}_t^{(k)} = q_{\boldsymbol{Y}_t^{(k)}}(\alpha)$, where $q_{\boldsymbol{Y}_t^{(k)}}(\alpha)$ is the level α empirical quantile of the vector of simulations

$$\boldsymbol{Y}_t^{(k)} = \begin{bmatrix} {}^{(1)}Y_t^{(k)} & \ldots & {}^{(B)}Y_t^{(k)} \end{bmatrix}$$

obtained from a bootstrap procedure. For the claimant k, the pseudo-response is

$$\widehat{C}_{T_3}^{(k)} = C_7^{(k)} + \sum_{t=8}^{13} \widehat{Y}_t^{(k)}.$$

For **Model D2**, **Model D3**, **Model D4** and **Model D5**, we select $\alpha = 0.6$, 0.7, 0.8 and 0.9, respectively.

The **Model E** is constructed exactly in the same way as **Model C3** but it uses prospective information about the 4 dynamic covariates available in the dataset. It is somehow analogous to **Model A** in the sense that it is not usable in practice. However, fitting this model allows to see if an additional model that would project censored dynamic covariates would be useful. In Table 4.3.1, we summarize the main specifications of the models.

Model	Training response vector	Covariates	Usable in practice?
Model A	$\{C_{13}\}$	x_7	No
Model B	$\{C_7^{(k)}\}_{k\in\mathcal{T}_B}$, $\mathcal{T}_B = \{k \in \mathcal{T} : T_3^{(k)} < 7\}$	x_7	Yes
Model C1	closed: $\{C_{T_3}\}$	x_7	Yes
	open: $\{\widehat{C}_{T_3} = \hat{\lambda}_j^c C_7\}$	x_7	
	($\hat{\lambda}$ from Equation 4.3.2)		
Model C2	closed: $\{C_{T_3}\}$	x_7	Yes
	open: $\{\widehat{C}_{T_3} = \hat{\lambda}_j^c C_7\}$	x_7	
	($\hat{\lambda}$ from bootstrap, quantile 0.8)		
Model C3	closed: $\{C_{T_3}\}$	x_7	Yes
	open: $\{\widehat{C}_{T_3} = \hat{\lambda}_j^c C_7\}$	x_7	
	($\hat{\lambda}$ from bootstrap, quantile 0.9)		
Model D1	closed: $\{C_{T_3}\}$	x_7	Yes
	open: $\{\widehat{C}_{T_3} = C_7 + \sum_{t=8}^{13} \widehat{Y}_t\}$	x_7	
	(with $\widehat{Y}_t = \widehat{\mu}_t$)		
Model D2	closed: $\{C_{T_3}\}$	x_7	Yes
	open: $\{\widehat{C}_{T_3} = C_7 + \sum_{t=8}^{13} \widehat{Y}_t\}$	x_7	
	(with $\widehat{Y}_t = q_{\boldsymbol{Y}_t}(0.6)$)		
Model D3	closed: $\{C_{T_3}\}$	x_7	Yes
	open: $\{\widehat{C}_{T_3} = C_7 + \sum_{t=8}^{13} \widehat{Y}_t\}$	x_7	
	(with $\widehat{Y}_t = q_{\boldsymbol{Y}_t}(0.7)$)		
Model D4	closed: $\{C_{T_3}\}$	x_7	Yes
	open: $\{\widehat{C}_{T_3} = C_7 + \sum_{t=8}^{13} \widehat{Y}_t\}$	x_7	
	(with $\widehat{Y}_t = q_{\boldsymbol{Y}_t}(0.8)$)		
Model D5	closed: $\{C_{T_3}\}$	x_7	Yes
	open: $\{\widehat{C}_{T_3} = C_7 + \sum_{t=8}^{13} \widehat{Y}_t\}$	x_7	
	(with $\widehat{Y}_t = q_{\boldsymbol{Y}_t}(0.9)$)		
Model E	closed: $\{C_{T_3}\}$	x_{13}	No
	open: $\{\widehat{C}_{T_3} = \hat{\lambda}_j^c C_7\}$	x_{13}	
	($\hat{\lambda}$ from bootstrap)		

Table 4.3.1: *Main specifications of XGBoost models. Unless otherwise stated, we have* $k \in \mathcal{T}$.

4.4 Results

From $\{\mathcal{D}_t^{(\mathcal{T})}\}_{t=1}^{7}$, which is the training dataset before evaluation date, it is possible to obtain a training run-off triangle by aggregating payments by accident and by development year, presented at Table 4.4.1.

	1	2	3	4	5	6	7
2004	79	102	66	49	57	48	37
2005	83	128	84	55	52	41	·
2006	91	138	69	49	38	·	·
2007	111	155	98	61	·	·	·
2008	100	178	99	·	·	·	·
2009	137	251	·	·	·	·	·
2010	155	·	·	·	·	·	·

Table 4.4.1: *Training incremental run-off triangle (in* \$100,000*).*

We can apply the same principle for the validation dataset $\mathcal{D}_{\mathcal{V}}$, which yields the validation run-off triangle displayed at Table 4.4.2.

	1	2	3	4	5	6	7	8+
2004	34	41	23	13	14	14	9	7
2005	37	60	36	29	45	21	20	24
2006	41	64	34	23	21	14	4	21
2007	46	67	40	37	15	18	3	13
2008	46	82	39	42	16	11	15	33
2009	54	109	62	51	31	36	11	2
2010	66	93	47	45	16	16	9	?

Table 4.4.2: *Validation incremental run-off triangle (in* \$100,000*).*

Based on the training run-off triangle, it is possible to fit many collective classical models (see [WM08] for an extensive overview). Once fitted, collective models are scored on the validation triangle. In the latter, data used to score models is displayed in black, as aggregate payments observed after the evaluation date are displayed in gray. Payments have been observed for 6 years after 2010, but this is not long enough for all claims to be settled. In fact, on December 31st 2016, 0.67% of claimants are associated with claims that are still open, mostly from accident years 2009 and 2010. Therefore, amounts in column "8+" for the most recent accident years in Table 4.4.2 are in fact too low. Based on available information, the total observed reserve amount is \$67,619,905 (summing all gray entries), but we can reasonably think that this amount would be closer to \$70,000,000 if we could observe more years.

Results for collective models are presented following two approaches:

- the Mack's model, for which we present results obtained with the bootstrap approach developed by [EV02] and based on both, quasi-Poisson and gamma

distributions;

- generalized linear models for which we present results obtained using a logarithmic link function and a variance function $\mathcal{V}(\mu) = \varphi\mu^p$ with $p = 1$ (quasi-Poisson), $p = 2$ (gamma), and $1 < p < 2$ (Tweedie).

For each model and based on the validation set, we present in Table 4.4.3 the expected value of the reserve, its standard error, as well as the 95% and the 99% quantiles of the predictive distribution of the total reserve amount. As it is generally the case, the choice of the distribution used to simulate the process error in the bootstrap procedure for the Mack's model has no significant impact on the results. Reasonable practices, at least in North America, generally require a reserve amount given by a high quantile (95%, 99% or even 99.5%) of the reserve's predictive distribution. Based on this, the reserve amount obtained by bootstrapping Mack's model is too high (between \$90,000,000 and \$100,000,000) compared to the observed value (approximately \$70,000,000). Reserve amounts obtained with generalized linear models are more reasonable (between \$77,000,000 and \$83,000,000), regardless of the choice of the underlying distribution. The predictive distribution for all collective models are shown in Figure 4.4.1.

In Table 4.4.3, we also present in-sample results, i.e., we use the same set of claimants to perform both estimation and validation. We note that the results are very similar, which tends to indicate some stability of the results obtained using these collective approaches.

			$\mathbb{E}$[Res.]	$\sqrt{\text{Var[Res.]}}$	$q_{0.95}$	$q_{0.99}$
Bootstrap	Quasi-Poisson	out-of-sample	76,795,136	7,080,826	89,086,213	95,063,184
Mack		in-sample	75,019,768	8,830,631	90,242,398	97,954,554
	Gamma	out-of-sample	76,803,753	7,170,529	89,133,141	95,269,308
		in-sample	75,004,053	8,842,412	90,500,323	98,371,607
GLM	Quasi-Poisson	out-of-sample	75,706,046	2,969,877	80,655,890	82,696,002
		in-sample	74,778,091	3,084,216	79,922,183	81,996,425
	Gamma	out-of-sample	73,518,411	2,263,714	77,276,416	78,907,812
		in-sample	71,277,218	3,595,958	77,343,035	80,204,504
	Tweedie	out-of-sample	75,688,916	2,205,003	79,317,520	80,871,729
	($\widehat{p} = 1.01$)	in-sample	74,706,050	2,197,659	78,260,722	79,790,056

Table 4.4.3: *Prediction results for collective approaches.*

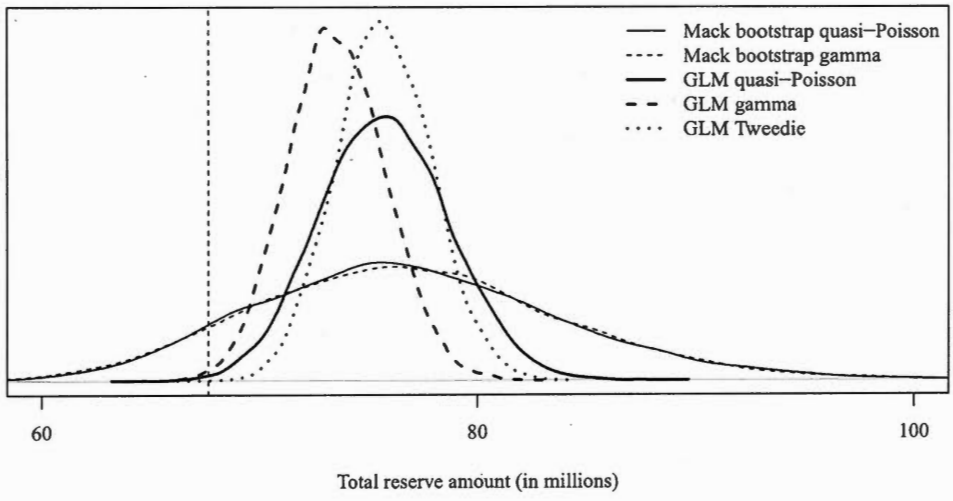

Figure 4.4.1: *Comparison of predictive distributions for collective models. The observed reserve amount is represented by the vertical dashed line.*

Individual models are trained on training set $\{\mathcal{D}_t^{(\mathcal{T})}\}_{t=1}^{7}$ and scored on validation set $\{\mathcal{D}^{(\mathcal{V})}\}_{t=8}^{13}$. In contrast to collective approaches, individual methods use micro-covariates and, more specifically, the reporting date. This allows us to distinguish between IBNR accidents and RBNS claims and, as previously mentioned, in this project we focus on the modeling of the RBNS reserve. Nevertheless, in our dataset, we observe very few IBNR accidents and therefore, we can reasonably compare the results obtained using both micro- and macro-level models with the observed amount ($67,619,905).

We consider the following approaches:

- individual generalized linear models, for which we present results obtained using a logarithmic link function and three variance functions: $\mathcal{V}(\mu) = \mu$ (Poisson), $\mathcal{V}(\mu) = \varphi\mu^p$ with $p = 1$ (quasi-Poisson) and $\mathcal{V}(\mu) = \varphi\mu^p$ with $1 < p < 2$ (Tweedie); and
- *XGBoost* models (**Model A**, **B**, **C1**, **C2**, **C3**, **D1**, **D2**, **D3**, **D4**, **D5** and **E**), described in Section 4.3.

Both approaches use the same covariates described in Section 4.2, which makes them comparable. For many claimants in both training and validation sets, some covariates are missing. Because generalized linear models cannot handle missing values, median/mode imputation have been performed for both training and validation sets. No imputation have been done for *XGBoost* models since missing values are processed automatically by the algorithm.

Results for individual GLM are displayed in Table 4.4.4, and predictive distributions for both quasi-Poisson and Tweedie GLM are shown in Figure 4.4.3. Predictive distribution for Poisson GLM is omitted since it is the same as the quasi-Poisson model, but with a much smaller variance. Based on our dataset, we

observe that the estimated value of the parameter associated with certain covariates is particularly dependent on the database used to train the model, e.g., in the worst case, for the quasi-Poisson model, we observe $\widehat{\beta}_{\text{Covariate 16 (modality 33)}} = 0.169$ (0.091) with the out-of-sample approach and $\widehat{\beta}_{\text{Covariate 16 (modality 33)}} = -1.009$ (0.154) with the in-sample approach. This can also be observed for many parameters of the model as shown in Figure 4.4.2 for the quasi-Poisson model. On this graph, we observe that, for most of the parameters, the values estimated on the validation set (grey dots) are inaccessible when the model is adjusted on the training set. In Table 4.4.4, we display results for both in-sample and out-of-sample approaches. As the results shown in Figure 4.4.3 suggest, there are significant differences between the two approaches. Particularly, the reserves obtained from the out-of-sample approach are too high compared to the observed value. Although it is true that in practice, the training/validation set division is less relevant for an individual generalized linear model because the risk of overfitting is lower, this suggests that some caution is required in a context of loss reserving. Given the presence of many zeros in the database, the fit of a generalized linear model with the Gamma is not good and the results have not been included in Table 4.4.4.

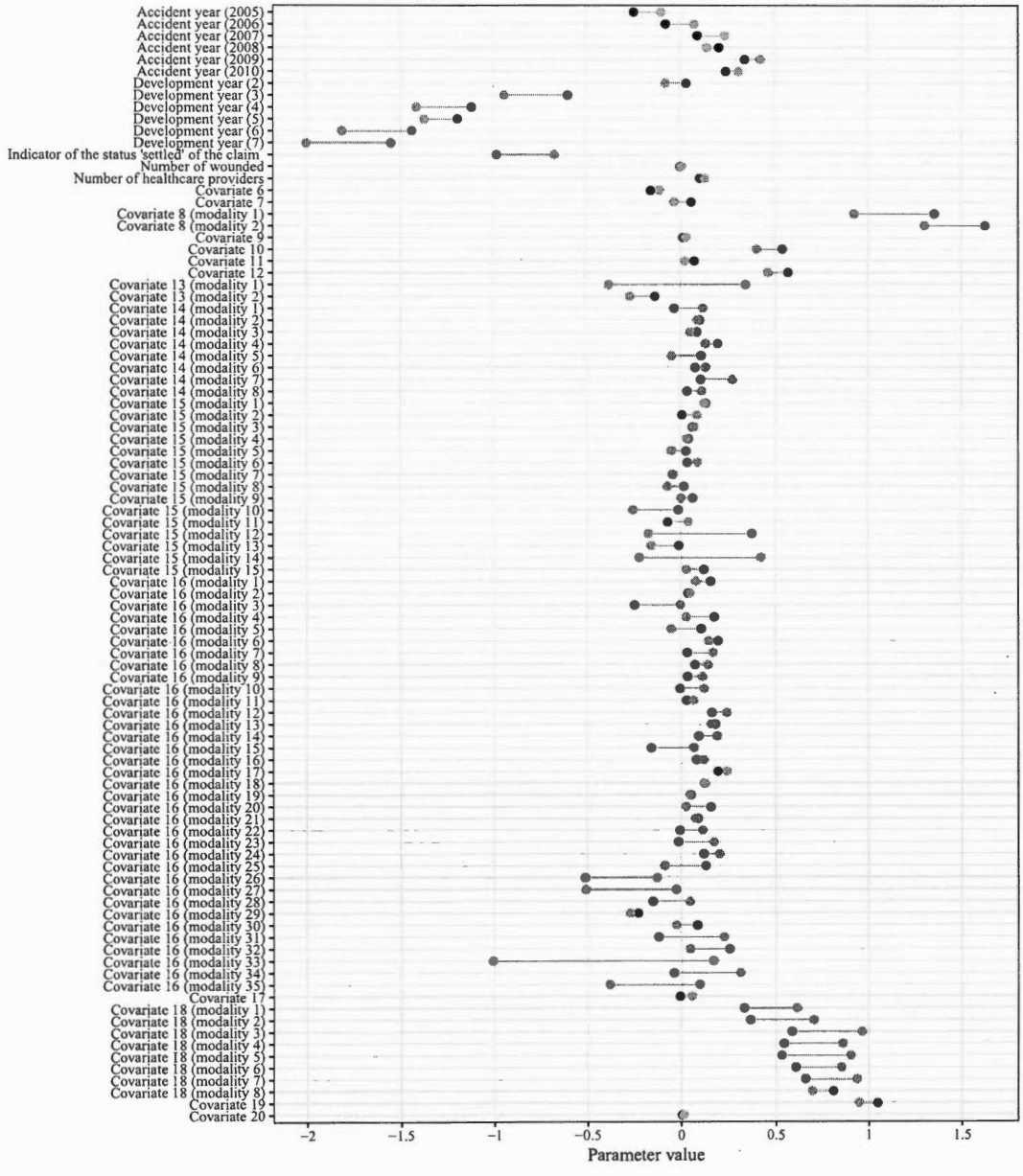

Figure 4.4.2: *Estimated parameters for quasi-Poisson individual GLM. The black dots correspond to the out-of-sample estimates, as the grey dot are the in-sample estimates.*

			$\mathbb{E}$[Res.]	$\sqrt{\text{Var[Res.]}}$	$q_{0.95}$	$q_{0.99}$
GLM	Poisson	out-of-sample	86,411,734	9,007	86,426,520	86,431,211
		in-sample	75,611,203	8,655	75,625,348	75,631,190
	Quasi-Poisson	out-of-sample	86,379,296	894,853	87,815,685	88,309,697
		in-sample	75,606,230	814,608	76,984,768	77,433,248
	Tweedie	out-of-sample	84,693,529	2,119,280	88,135,187	90,011,542
		in-sample	70,906,225	1,994,004	74,098,686	75,851,991

Table 4.4.4: *Prediction results for individual generalized linear models using covariates.*

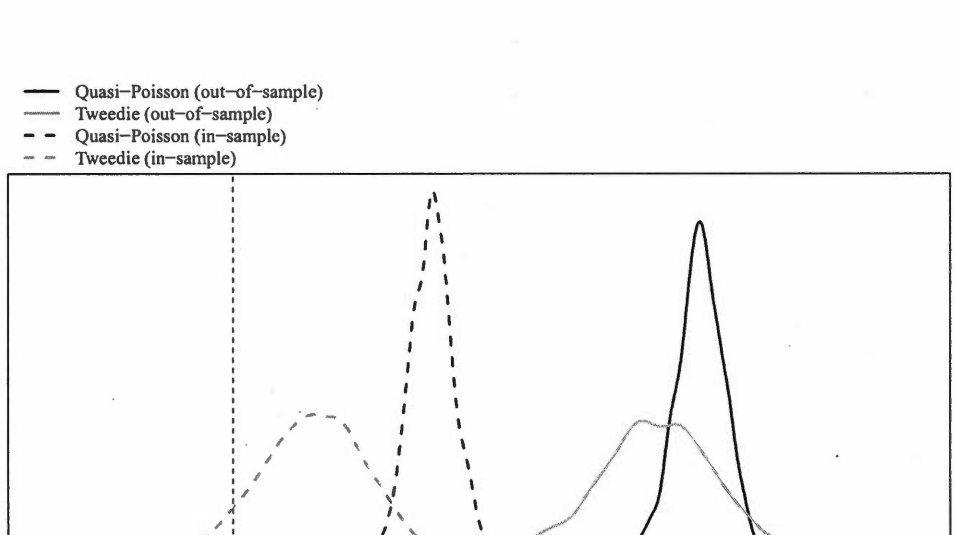

Figure 4.4.3: *Predictive distributions for in-sample and out-of-sample individual GLM with covariates.*

Results for *XGBoost* models are displayed in Table 4.4.5. For all models, the learning rate ν in Equation 3.2.13 is around 10 %, which means our models are quite robust to overfitting. We use a maximum depth of 3 for each tree. A higher value would make our model more complex but also less robust to overfitting. All

those hyperparameters are obtained by cross-validation. Notice that in this work, cross-validation is only used for hyperparameter tuning: model assessment is done by separating the database into training and validation sets.

		$\mathbb{E}$[Res.]	$\sqrt{\text{Var[Res.]}}$	$q_{0.95}$	$q_{0.99}$
XGBoost	**Model A**	73,204,228	3,742,810	79,329,916	82,453,032
	Model B	14,339,470	6,723,608	25,757,061	30,643,369
	Model C1	60,953,413	2,379,841	64,946,493	66,890,092
	Model C2	67,655,960	2,411,739	71,708,313	73,762,242
	Model C3	71,104,515	2,410,433	75,069,298	77,144,866
	Model D1	64,436,000	3,819,808	71,030,238	73,873,276
	Model D2	36,605,131	3,959,622	45,215,241	43,377,677
	Model D3	47,637,431	4,034,744	54,277,446	57,648,568
	Model D4	68,313,731	4,176,418	75,408,868	78,517,966
	Model D5	122,893,765	4,265,776	130,210,785	133,208,866
	Model E	67,772,822	2,387,476	71,722,744	73,540,516

Table 4.4.5: *Prediction results for individual approaches using covariates.*

Not surprisingly, we observe that the **Model B** is completely off the mark, underestimating the total reserve by a large amount. It confirms that the selection bias, at least in this example, is real and substantial.

Model C1, **Model C2** and **Model C3** consider a collective model, i.e., without micro-covariates, to create pseudo-responses and then, use all micro-covariates available in order to predict final paid amounts. It seems that using the expected value as a pseudo-response (**Model C1**) is not a conservative approach. With

a slightly lower expectation and variance, **Model C2** and **Model C3** are quite similar to **Model A**. Since the latter uses real responses for its training, this is an indication that the method used for claim development is reasonable. In addition, the 99^{th} percentile of **Model C3**'s distribution is above the observed sum of payments by a fair amount. This is a good thing since it means the amount of money reserved by the insurance company will be plenty most of the time. We also do not want the 99^{th} percentile to be too high above the observed amount, because it would mean a loss of investment income. **Model C3** is more conservative than **Model C2** since it uses a higher quantile of the distribution of the development factors for claim development. Hence, the choice of the quantile for claim development relies on the insurance company, and should be chosen for instance with cross-validation. The predictive distributions for **Model A**, **Model C1**, **Model C2** and **Model C3** are displayed in Figure 4.4.4.

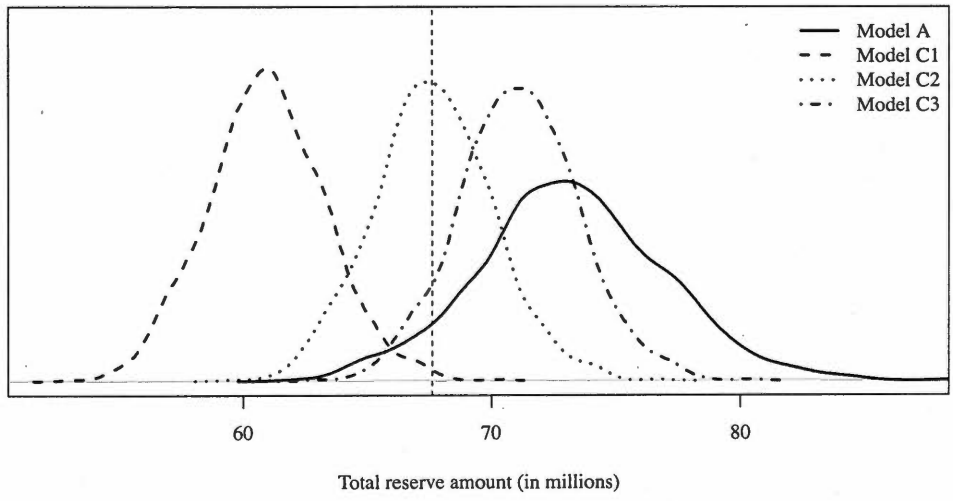

Figure 4.4.4: *Predictive distributions for* XGBoost ***Model A***, ***C1***, ***C2***, *and* ***C3***.

In Figure 4.4.5, we compare the predictive distributions of **Model A**, **Model D1**, **Model D2**, **Model D3** and **Model D4**. The predictive distribution of the **Model D1** is not shown because it is very similar to that of the **Model D3**. All **D** models use a quasi-Poisson GLM for claim development, but each one with a different quantile of the quasi-Poisson distribution. As **Model C2** and **Model C3**, these are almost perfect translations of each other. We note that the results obtained are very unstable, ranging from \$35,000,000 when a 60^{th} percentile is used for pseudo-responses (**Model D2**) to nearly \$100,000,000 when a 90^{th} percentile is used (**Model D5**). The standard deviation is also very high, close to \$5,000,000 in comparison with that of the **C** models. On one hand, the insurance company must choose a confidence level that does not compromise its solvency as do **Model D**. On the other hand, the quantile chosen should not be too high, at the risk of losing investment income, as does **Model D5**. Here, the best model seems to be **Model D4**, suggesting to choose the 80^{th} percentile.

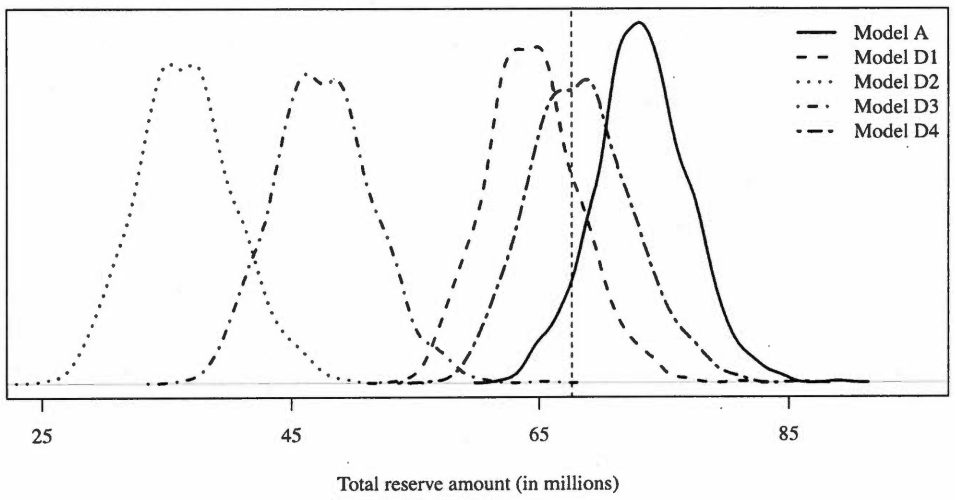

Figure 4.4.5: *Predictive distributions for* XGBoost ***Model A, D1, D2, D3 and D4.***

Model E is identical to **Model C3** with the exception of dynamic variables whose value at the evaluation date was artificially replaced by the ultimate value. We note that, at least in this case study, the impact is negligible (see Figure 4.4.6). There would be no real interest in building a hierarchical model that allows, first, to develop the dynamic variables and, second, to use an *XGBoost* model to predict final paid amounts. We have not tested whether the replacement of dynamic variables by their ultimate value would make a significant change for other than **Model C3**. However, since the models are similar, it is assumed that the results would be similar.

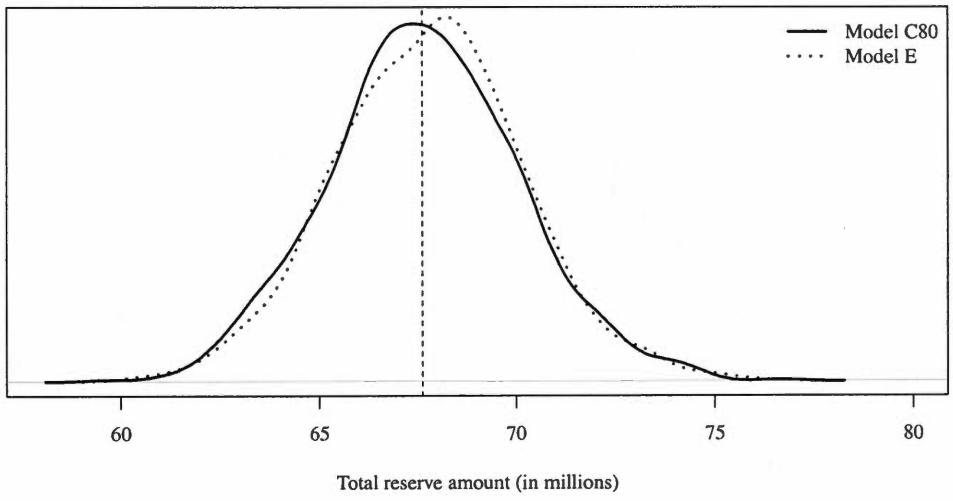

Figure 4.4.6: *Comparison of predictive distributions for* ***Model E*** *and* ***Model C3****. The observed total paid amount is represented by the vertical dashed line.*

Notice that in this paper, we always consider predictive distributions to compare models. One might wonder why we do not use criteria often used in machine learning as the Root Mean Squared Error (RMSE) or the Mean Absolute Error

(MAE). The reason lies, at least in part, in the unbalancing of the data. Indeed, the database used in this work contains a lot small claims (many of which are closed at zero) and very few big claims. Therefore, because they are symmetric error functions, RMSE and MAE favor models that predict low reserves. At the limit, we could think that a model predicting a RBNS reserve of exactly zero for each claim would have a good RMSE or MAE since the observed reserve is zero for a big proportion of claimants. However, we know that it would be a poor model since the sum of the predicted reserves would be zero!

CONCLUSION

This paper studies the modeling of loss reserves for a property and casualty insurance company using micro-level approaches. More specifically, we apply generalized linear models and gradient boosting models designed to take into account the characteristics of each individual claimant, as well as individual claim. We compare those models to classical approaches and show their performance on a detailed dataset from a Canadian insurance company. The choice of a gradient boosted decision trees model is motivated by its great performance for prediction on structured data. Also, this type of algorithm requires very little data preprocessing, which is a great benefit. Among all existing gradient boosting algorithms, *XGBoost* have been chosen, especially for its relatively short calculation time.

Through a case study, we have mainly shown that

(1) the use of a micro-level model based solely on generalized linear models could be unstable for loss reserving and

(2) an approach combining a macro-level model for the artificial completion of open claims and a micro-level gradient-boosting model could be an interesting approach for an insurance company.

In addition, we illustrate that the censored nature of the data could strongly bias the results and we propose some solutions such as projecting total paid amount of non settled claims using different methods such as bootstrap chain-ladder and generalized linear models.

The gradient boosting models presented in this paper allow to compute a prediction for the total paid amount of each claimant. However, an insurer could also be interested to model the payment schedule, namely to predict the date and the amount of each individual payment. We know that payments for insured belonging to the same claim aren't independent: it is well known that claim amounts for claimants of a same claim are positively correlated. Therefore, one could extend the model by adding a dependence structure between claimants. The same principle could be applied with the different types of coverage (medical and rehabilitation, income replacement, etc.). Dynamic covariates can change over time, which makes their future value random. In this work, we assumed that their value will not change after the evaluation date and we checked that the impact is marginal. However, for a different database, this could have a significant impact on the results. A possible refinement would be to build a hierarchical model that first predicts the ultimate values of dynamic covariates before inputting them into the gradient boosting algorithm. In the validation part, models were penalized equally regardless of the accident year of the claim. Because it is even more crucial to estimate reserves accurately for recent accident years, some kind of importance sampling could be used in order to penalize models more severely when they are wrong about recent claims. Finally, we could consider another criterion to evaluate the performance of models. In this paper, we look at the predictive distributions and evaluate by eye if the model is good or not. We said earlier that symmetric error functions as the RMSE and the MAE are poor criteria due to the unbalanced data. It could therefore be interesting to look for another error function that penalizes more errors on big claims compare to small ones. That would allow us to evaluate the models in a more objective way.

BIBLIOGRAPHY

[AP14] Antonio, K., & Plat, R. (2014). Micro-level stochastic loss reserving for general insurance. *Scandinavian Actuarial Journal, 2014*(7), 649–669.

[AR89] Arjas, E. (1989). The claims reserving problem in non-life insurance: Some structural ideas. *ASTIN Bulletin, 19*(2), 139–152.

[BR17] Baudry, M., & Robert, C. Y. (2017). Non-parametric individual claim reserving in insurance. *Working paper.*

[BE86] Benjamin, S., & Eagles, L. M. (1986). Reserves in Lloyd's and the London market. *Journal of the Institute of Actuaries, 113*(2), 197–256.

[BF72] Bornhuetter, R. L., & Ferguson, R. E. (1972). The Actuary and IBNR. *In Proceedings of the Casualty Actuarial Society, 59*(112), 181–195

[BR84] Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). Classification And Regression Trees. *Wadsworth.*

[BU80] Bühlmann, H., Schnieper, R., & Straub, E. (1980). Claims reserves in casualty insurance based on a probabilistic model. *Bulletin of the Association of Swiss Actuaries, 80*, 21–45.

[CP16] Charpentier, A., & Pigeon, M. (2016). Macro vs. micro methods in non-life claims reserving (an econometric perspective). *Risks, 4*(2), 12.

[CG16] Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

[CO83] Copas, J. B. (1983). Regression, prediction and shrinkage. *Journal of the Royal Statistical Society*, 311–354.

[DE09] Debye, P. (1909). Der lichtdruck auf kugeln von beliebigem material. *Annalen der physik, 335*(11), 57–136.

[EV02] England, P. D., & Verrall, R. J. (2002). Stochastic claims reserving in general insurance. *British Actuarial Journal, 8*(3), 443–518.

[FS97] Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences, 55*(1), 119–139.

[FH01] Friedman, J., Hastie, T., & Tibshirani, R. (2001). The elements of statistical learning (Vol. 1, No. 10). *New York: Springer series in statistics.*

[FR01] Friedman, J. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics, 29*(5), 1189–1232.

[HA96] Haastrup, S., & Arjas, E. (1996). Claims reserving in continuous time; a non-parametric Bayesian approach. *ASTIN Bulletin, 26*(2), 139–164.

[HA80] Hachemeister, C.A. (1980). A stochastic model for loss reserving. *Transactions of the* 21^{st} *International Congress of Actuaries*, 185–194.

[HE94] Hesselager, O. (1994). A Markov model for loss reserving. *ASTIN Bulletin 24*(2), 183–193.

[HI16] Hiabu M., Margraf, C., Martínez-Miranda, M.D., Nielsen, J.P. (2016). The link between classical reserving and granular reserving through double chain ladder and its extensions. *British Actuarial Journal, 21*(1), 97–116.

[HU15] Huang, J., Qiu, C., & Wu, X. (2015). Stochastic loss reserving in discrete time: individual vs. aggregate data models. *Communications in Statistics - Theory and Methods, 44*, 2180–2206.

[JE89] Jewell, W.S. (1989). Predicting IBNYR events and delays. *ASTIN Bulletin, 19*(1), 25–55.

[JF13] Jin, X., & Frees, E. W. J. (2013). Comparing micro- and macro-level loss reserving models. *Presentation at ARIA.*

[KI94] Kirkegaard, T. (1994). Praktisk anvendelse af en reservemodel pa ulykkesforsikringer. *Master thesis, University of Copenhagen.*

[LA07] Larsen, C. (2007). An individual claims reserving model. *ASTIN Bulletin, 37*(1), 113–132.

[ST03] Simon, J., & Thoma, M. (2003). Lecture notes in control and information sciences. *Springer*, 59 – 78.

[LM16] Lopez, O., Milhaud, X., & Thérond, P. E. (2016). Tree-based censored regression with applications in insurance. *Electronic journal of statistics, 10*(2), 2685–2716.

[MA93] Mack, T. (1993). Distribution-free calculation of the standard error of chain-ladder reserve estimates. *ASTIN Bulletin, 23*(2), 213–225.

[MN89] McCullagh, P., & Nelder, J. A. (1989). Generalized linear models. *CRC press.*

[NO86] Norberg, R. (1986). A contribution to modeling of IBNR claims. *Scandinavian Actuarial Journal, 1986*(3–4), 155–203.

[NO93A] Norberg, R. (1993). Prediction of outstanding liabilities in non-life insurance. *ASTIN Bulletin, 23*(1), 95–115.

[NO93B] Norberg, R. (1993). Prediction of outstanding liabilities: parameter estimation. *Proceedings of the XXIV ASTIN Coll.,* 255–266.

[NO99] Norberg, R. (1999). Prediction of outstanding liabilities. II Model variations and extensions. *ASTIN Bulletin, 29*(1), 5–25.

[PA13] Pigeon, M., Antonio, K., & Denuit, M. (2013). Individual loss reserving with the multivariate skew normal framework. *ASTIN Bulletin, 43*(3), 399–428.

[QM04] Quarg, G., & Mack, T. (2004). Munich chain ladder. *Blätter der DGVFM, 26(4),* 597-630.

[TM08] Taylor, G., McGuire, G., & Sullivan, J. (2008). Individual claim loss reserving conditioned by case estimates. *Annals of Actuarial Science, 3*(1-2), 215–256.

[WM08] Wüthrich, M., & Merz, M. (2008). Stochastic claims reserving methods in insurance. *Wiley Finance.*

[WU18] Wüthrich, M. V. (2018). Machine learning in individual claims reserving. *Scandinavian Actuarial Journal,* 1–16.

[ZZ09] Zhao, X., Zhou, X., & Wang, J. (2009). Semiparametric model for prediction of individual claim loss reserving. *Insurance: Mathematics and Economics, 45*(1), 1–8.

[ZZ10] Zhao, X., & Zhou, X. (2010). Applying copula models to individual claim loss reserving methods. *Insurance: Mathematics and Economics, 46*(2), 290–299.