Stockholms universitet

Gradient Boosted Trees Applied to Chain-Ladder Reserving

Fredrik Käll

Masteruppsats i försäkringsmatematik
Master Thesis in Actuarial Mathematics

Masteruppsats 2022:13
Försäkringsmatematik
September 2022

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Mathematical Statistics
Stockholm University
Master Thesis **2022:13**
http://www.math.su.se

Gradient Boosted Trees Applied to Chain-Ladder Reserving

Fredrik Käll*

September 2022

Abstract

This thesis investigates whether non-life claims reserving can be improved by using more information regarding each claim and machine learning techniques. As a foundation, Wühtrich's article *Neural Networks applied to Chain-Ladder reserving* has been used, with the modification that Gradient Boosted Trees have been used instead of Neural Network. We begin by obtaining a model by walking through the fitting process. The model is then used to predict the outstanding reserves and compared to Mack's Chain-Ladder predictions. Further, a comparison of the two models MSEP is made to investigate the variation of the two models. The comparison shows that the Gradient Boosted Trees perform as well as Chain-Ladder for earlier, more developed years. However, in later years, the performance is not as good. We end the thesis with a discussion on why the boosted trees did not perform well and what could be done to improve the predictions.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: fredrik.kall@outlook.com. Supervisor: Mathias Lindholm.

Acknowledgement

I would like to thank Mathias Millberg Lindholm for feedback and valuable discussions. I would also like to thank Adam Pettersson and Dennis Uygur Andersson for their work as sounding boards. And finaly, I would like to thank my cohabit Hedda Hultgren and our two cats for their support.

Contents

1 Introduction

When an accident occurs, it does not necessarily get reported directly. There are several reasons why this is the case. For example, an expert opinion is needed to verify that it is a particular injury, or it could be that it takes time even to realize that an accident has happened.

Since there are rules and demands on how an insurance company should fulfill their future commitments, there are often two types of payments mentioned within insurance, Incurred But Not Reported (IBNR) and Reported But Not Settled (RBNS). We will give a brief explanation of both with an example.

Example:
Suppose an accident occurs at time T_A, is reported at time T_R and closed at time T_C. Further, assume that the accident results in three payments, p_1, p_2 and p_3. We then have $T_A < T_R < p_1 < p_2 < p_3 < T_C$.
Given that we are now located at time τ, every timepoint $< \tau$ is known.

Scenario 1: If $\tau < T_R$ we are not aware that this accident has happened, we, therefore, classify the payments as IBNR

Scenario 2: If $p_1 < \tau < p_2$ the accident is known and we have paid a part of it. However p_2 and p_3 have not yet been paid, and are therefor classified as RBNS

The most common models used today for estimating IBNR are Bornhuetter-Furgesson (BF) and Chain-Ladder (CL). BF and CL are popular because they are both easy to compute and assume that the claims are distribution-free but still give a good estimate of the total claims cost, also called Ultimo. However, as the development proceeds, there are more possibilities to apply, for example, machine learning techniques.

There are multiple machine learning techniques and claims reserving models, hence multiple combinations that can be used. Machine learning can also be used to fit the original model, or it can be used to improve the original model by creating adjustments based on the residuals. Further, by using machine learning, there is a possibility to include more information about each claim than the original model allows.

In this thesis, we will use Wüthrich's article *Neural Networks applied to Chain-Ladder reserving* [10] as a foundation. However, we will use Gradient Boosting Machines instead of Neural Networks.

2 Methods

In this section we take a look at the theory that is used within this thesis.

2.1 Chain-Ladder

As the introduction mentions, the Chain-Ladder method is one of the most common models to estimate IBNR. CL aims to estimate the ultimo for a specific month, quarter, or year period. All the theory below is from [6] and [7]

Let's denote the cumulative claim cost for development year k and accident year i as C_{ik}, where $1 < i < I$ and $1 < k < K$. We can then represent the Ultimo for accident year i as C_{iK}. The Chain-Ladder can be presented as a triangle, Figure 1. The goal is to estimate the lower right triangle, marked as gray.

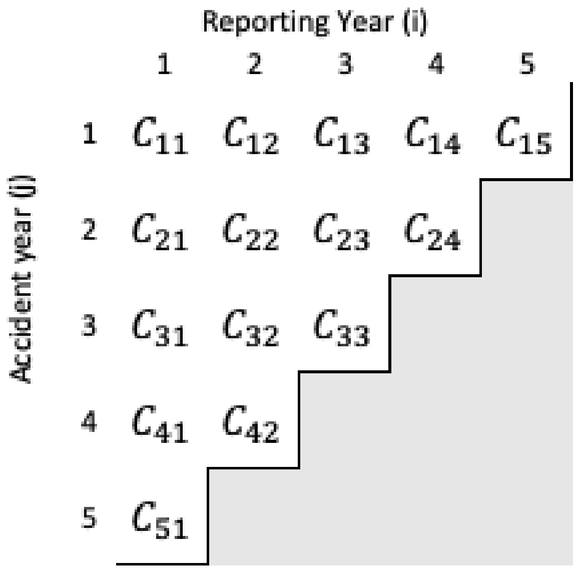

Reporting Year (i)
1
2
3
4
5
Accident year (j)
1
2
3
4
5
C_{11} C_{12} C_{13} C_{14} C_{15}
C_{21} C_{22} C_{23} C_{24}
C_{31} C_{32} C_{33}
C_{41} C_{42}
C_{51}

Figure 1: Chain-Ladder illustrated as a triangle

When working with Chain-Ladder there are three assumptions made. The first assumption is that the expected value of $C_{i,k+1}$ is a factor, f_k times the previous periods value, C_{ik}. This can be expressed as

$$\mathrm{E}(C_{i,k+1} \,|\, C_{i,1}, ..., C_{i,k}) = f_k C_{i,k} \text{ for } 1 \leq i \leq I, 1 \leq k \leq K-1.$$

Assumption two is that two different accident years are independent i.e.

$$\{C_{j1}, C_{j2}, \ldots, C_{jK}\}, \{C_{i1}, C_{i2}, \ldots, C_{iK}\}, j \neq i \text{ are independent}, \forall\ i, j$$

The third and last assumption is there exist a constant, σ_k, such that

$$\mathrm{Var}(C_{i,k+1}|C_{i,1},...,C_{ik}) = C_{ik}\sigma_k^2$$

Given these three assumptions we can state the model

$$C_{i,k+1} = C_{ik}f_k + \sigma_k\sqrt{C_{ik}}\epsilon_{i,k+1} \tag{1}$$

where

$$\mathrm{E}[\epsilon_{i,k+1}|C_{i,1},...,C_{ik}] = 0, \quad \mathrm{Var}(\epsilon_{i,k+1}|C_{i,1},...,C_{ik}) = 1 \text{ for } 1 \leq i \leq I, 1 \leq k \leq K-1.$$

$\hat{f}$ can then be estimated using least squared, given k, by

$$\hat{f}_k = \frac{\sum_{j=1}^{J-k} C_{j,k+1}}{\sum_{j=1}^{J-k} C_{j,k}}$$

and we can then write the estimation of Ultimo as

$$\hat{C}_{iJ} = C_{i,K-i} \cdot \hat{f}_{K-1} \cdot ... \cdot \hat{f}_{K-i}$$

2.2 Tree-Based Gradient Boosting Machines

2.2.1 Regression Trees

Decision trees can be used for either regression or classification. The underlying mathematics is the same for both; however, we will use regression trees in this thesis.

The main idea of decision trees is that you split your features space into J non-overlapping regions, R_1, ..., R_J. Given a new input vector, it is then possible to classify the output depending which region it falls into. This can be expressed as Equation (2).

$$h(\mathbf{x}_i) = \sum_{m=1}^{J} c_m I(\mathbf{x}_i \in R_m) \tag{2}$$

where $I(\mathbf{x}_i \in R_m)$ is a indicator which equals 1 if $\mathbf{x}_i$ belongs to region R_m and 0 otherwise.
The goal when creating a tree is then to minimize the residual sum of squares, RSS, given by

$$\sum(y_i - h(\mathbf{x}_i))^2.$$

Minimizing this results in that the best estimate for $\hat{c}_m$ is the average of y_i in region R_m, i.e.

$$\hat{c}_m = \mathrm{ave}(y_i \mid \mathbf{x}_i \in R_m)$$

However, since it is impossible to consider all possible partitions, a so called greedy method is used when creating a tree. The greedy method is a top-down method, meaning that we start with the root and then work our way down. We start by dividing the predictor space into two regions given predictor j and splitting point s.

$$R_1(j,s) = \{X \mid X_j \leq s\} \text{ and } R_2(j,s) = \{X \mid X_j > s\}$$

The next step is then to choose which j and s that minimize the RSS, i.e solving Equation (2) .

$$\min_{j,s}\left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2\right] \tag{3}$$

When the j and s that minimize RSS are found, the same procedure is then repeated until a criterion is fulfilled, e.g., a maximum of observations in each region. It is worth mentioning that when using the greedy method, the same feature is possibly used in multiple depths of the tree.

2.2.2 Gradient Tree-Boosting

When using weak classifiers, such as trees, they are often improved by *boosting*. The idea behind boosting is to combine multiple weak models into one. There are multiple tree boosting techniques, for example, Random Forest, AdaBoost, and Gradient Boosting, where the last one is used in this thesis.

Gradient Tree-Boosting is an ensemble of many trees where you start with a root and then add new trees recursive. The approach is to start with an initial guess. We then use a loss function, $L(y_i, f(x_i))$, that is optimized using gradient descent. The gradient is defined as

$$g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

The aim is then to create a new tree that fits the values obtained from the gradient.

We now have a model that consists of the initial guess and a tree. This process is then repeated a predefined iterations, M. When all M trees are done we end up with a model defined as

$$\hat{f}(x) = T_0(x,\theta) + \sum_{k=1}^{M} T_k(x,\theta)$$

This process can be written down more broadly, as in Algorithm 1, below.

Algorithm 1 Gradient Tree Boosting Algorithm

```
1. Initialize $g_0(x) = \arg\min_\gamma \sum_{i=1}^N L(y_i, \gamma)$
2. For $m = 1$ to $M$:
    (a) For $i$ = 1, 2,..., N compute
        $$r_{im} = -\left[\frac{\partial L(y_i, g(x_i))}{\partial g(x_i)}\right]_{g=g_{m-1}}$$
    (b) Fit a regression tree to the targets $r_{im}$ giving terminal regions $R_{jm}$, $j = 1, 2, ...J_m$
    (c) For $j$ = 1,2,...,$J_m$ compute
        $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(y_i, g_{m-i}(x_i) + \gamma)$$
    (d) Update $g_m(x) = g_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
3. Output $\hat{g}(x) = g_M(x)$
```

As can be seen in Algorithm 1 two parameters can be tuned, the number of iterations, M, and the size of each tree, J_m. The question is then how to select proper values for these parameters.

We start by examining how to choose a proper J_m. As mentioned in Section 2.2.1 big trees tend to over-fit, which results in high variance. Further, since boosting uses multiple trees, this may result in increased computation. The simplest way to avoid this is to set $J_m = J, \forall m$, i.e., restricting the trees to the same size.

There are two ways to control the impact of M in boosting, either find the best suited M or by using shrinkage. When using shrinkage there are multiple ways to implement this. The technique that will be used in this thesis is to scale the contribution of each tree by a factor $0 < \eta < 1$, i.e. replace line 2(d) in Algorith 1 with

$$g_m(x) = g_{m-1}(x) + \eta \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm}).$$

We refer to [4] for further reading regarding shrinkage.

2.3 Mean Square Error of Prediction (MSEP)

To see how well a prediction is, MSEP is often used, which is defiened as

$$\text{MSEP}(\hat{C}_{iI}) = \text{E}((\hat{C}_{iI} - C_{iI})^2).$$

However, since we are using Chain-Ladder and different accident contains different amount of information, it is more convenient to use the Conditional MSEP,

$$\text{MSEP}(\hat{C}_{iI}|D) = \text{E}((\hat{C}_{iI} - C_{iI})^2|D)$$

where D $= \{C_{ik} \,|\, i + k \leq I\}$, i.e., the set of data observed so far.

This can be rewritten as

$$\text{MSEP}(\hat{C}_{iI}|D) = \text{Var}(C_{iI}|D) + (\text{E}(C_{iI}|D) - \hat{C}_{iI})^2.$$

Under the assumptions that are made in section 2.1 the estimation of MSEP can be rewritten as

$$\widehat{\text{MSEP}(\hat{C}_{iI}|D)} = \hat{C}_{iI}^2 \sum_{k=I+1-i}^{I-1} \frac{\hat{\sigma}_k^2}{\hat{f}_k^2} \left(\frac{1}{\hat{C}_{ik}} + \frac{1}{\sum_{j=1}^{I-k} C_{jk}} \right)$$

where $\hat{\sigma}^2$ is the unbiased estimatior of σ^2 and is defiened as

$$\hat{\sigma}_k^2 = \frac{1}{I-k-1} \sum_{i=1}^{I-k} C_{ik} \left(\frac{C_{i,k+1}}{C_{ik}} - \hat{f}_k \right)^2$$

For the proof we refer to [6].

As can be seen, there arise issues when $k = I - 1$. This can be solved either by assuming that the claim cost is fully developed after $I - 1$ years, i.e. $\hat{f}_{I-1} = 1$, which will give us that $\hat{\sigma}_{I-1}^2 = 0$. If we do not assume that it's fully developed, we could instead extrapolate the usually exponential decreasing series $\hat{\sigma}_1^2, \hat{\sigma}_2^2, ..., \hat{\sigma}_{I-2}^2$. The extrapolation could either be a log linear regression or as simple as

$$\frac{\hat{\sigma}_{I-3}^2}{\hat{\sigma}_{I-2}^2} = \frac{\hat{\sigma}_{I-2}^2}{\hat{\sigma}_{I-1}^2}$$

which holds true as long as $\hat{\sigma}_{I-3}^2 > \hat{\sigma}_{I-2}^2$. Given the last constraint we can use

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

2.4 Bias-Variance Trade Off

When creating a model, the goal is to minimize the expected error on unseen data. The risk that arises is an overfitted model and hence too complex. A complex model might predict the training data well, but not the predicted one. This results in a model with high variance. On the other hand, a simple model does not use all the information, which results in a bad prediction, i.e., high bias.

Let's assume that we expect that the response variable $Y = h(x) + \epsilon$, where

$E(\epsilon) = 0$. The expected prediction error, MSE, for an observation can then be written as

$$\mathrm{Err}(x_0) = \mathrm{E}\left(y_0 - \hat{h}(x_0)\right)^2 = \mathrm{Var}\left(\hat{h}(x_0)\right) + \left[\mathrm{Bias}\left(\hat{h}(x_0)\right)\right]^2 + \mathrm{Var}(\epsilon)$$

The goal is then to find a model that minimize the expected prediction error. An illustration of the MSE can been seen in Figure 2 below

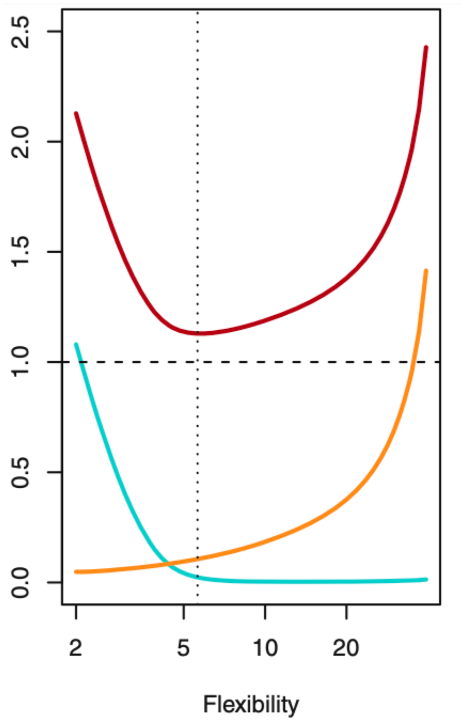


Figure 2: Squared bias (blue curve), variance (orange curve), $\mathrm{Var}(\epsilon)$ (dashed line) and MSE (red curve). Figure taken from [5].

2.5 k-fold Cross Validation

The best way to find the model that minimizes the MSE is to use one dataset for training and one for validation. However, there are often times data for validation is not available. When this is the case, resampling methods can be handy.

The idea of resampling methods is to use the training data for training and validation. This is achieved by splitting the data into training and validation data. There are multiple different methods available for resampling, but the one used in this thesis is k-fold cross validation.

Given a data set of n observations, top row in Figure 3, we can divide it into k

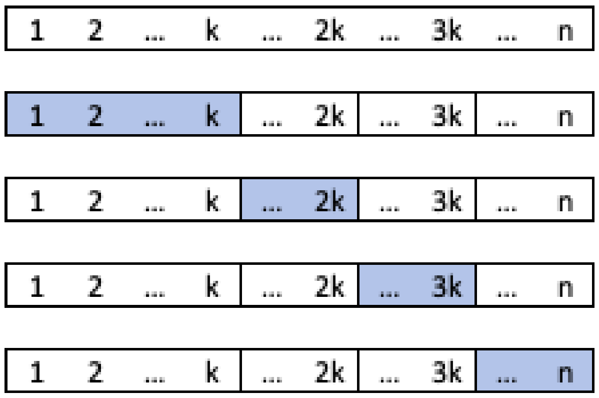


Figure 3: A 4-fold illustrated. The top row is the whole data. The blue-marked are the parts that are left out when calibrating the model based on the data for a specific fold (row).

parts. Figure 3 illustrates a division of 4 parts. We are then able to use $\frac{(k-1)}{k}$ parts of the data, represented in white, to create a model and then validate the model using the data that have been left out, blue. This is done by using

$$\text{MSE} = \sum_{i \in \mathcal{I}_{(k)}} (y_i - \hat{h}(x_i))^2$$

where $\mathcal{I}_{(k)}$ are all the observations in part k, i.e., the left out part. This is then repeated for all combinations. The cross validation can then be calculated as the mean of all the MSEs, i.e.

$$\text{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i$$

We can then compare the cross-validation of different models and choose the model that obtained the smallest value.

3 Model

As mentioned in the introduction, one of the most common models to estimate IBNR is the Chain-Ladder due to the fact that it is easy to compute and assume that the claims are distribution-free. There is, however, one drawback with Chain-Ladder, it assumes that the data is homogeneous.

The scope of this thesis is to extend the regular Chain-Ladder by allowing the development factors to be dependent on more specific claims data. This is done by extending the model to be dependent on a feature vector $\mathbf{x} \in \mathcal{X}$. We can then use sub-portfolios to estimate our development factors, which can be used to calculate the ultimo.

As the original Chain-Ladder, this can be seen as a regular linear regression. By dividing the data into sub-portfolios, we are able to use different methods rather than the least squared. As previously mentioned, Gradient Boosting Machines will be used in this thesis, but other techniques can be used, e.g., support vector machines.

As mentioned in the introduction, a similar approach has previously been used by Wüthrich [10] where neural networks have been used for fitting the development factors. The same approach will be used in this thesis, with the modification that GBMs will be used instead of Neural Networks. There are a few advantages with Boosted Trees (BT) compared to Neural Networks (NN). First off, trees are much more straightforward than NN. Trees are built using statements, e.g., are there more than two rooms in the apartment, while NN uses a combination of matrix multiplication and non-linear functions. Hence, BTs are more transparent.

The second advantage of BTs is the acceptance of input. Since the BTs use statements to find the best fit, both categorical and continuous variables are allowed. NN on the other hand, which uses matrix multiplication and non-linear functions, does not allow categorical input, and the continuous variables need to be scaled. There are, however, solutions for these issues, but depending on the data, this can be time-consuming.

The two advantages above often result in a faster computation for BTs than NNs.

The data we will be using is simulated using a simulation machine [1]. The data, where the first ten rows are presented in Table A.1, consists of 2'502'078 observations, where each observation consists of information regarding

- The Line of Business (LoB) $\in \{1, 2, 3, 4\}$
- Claims Code (cc) $\in \{1, 2, ..., 53\}$

- Accident Quarter (AQ) $\in \{1, 2, 3, 4\}$
- Age of the injured (age) $\in \{15, 16, .., 70\}$ in years when claims occurrence
- Injured body part (inj_part) $\in \{10, 11, .., 99\}$

In this thesis, all features above will be used, which gives us $|X| = 4{\cdot}4{\cdot}53{\cdot}56{\cdot}90 = 4'273'920$ different possible feature values. These features can be seen as static, i.e., they do not change over time. The distribution of each feature is presented in Figure A.2. What can be seen in the figure is that the claims are uniformly distributed over accident year and accident quarter. The figure, however, presents the distribution of all generated observations. Since the scope of this thesis is to predict the ultimo, we could expect that the distribution of the "known" data looks a bit different. We can also see that the number of reported claims decreases for higher ages. There are also a few claim codes and injured body parts that have fewer reported claims. This might cause problems when the ultimo for features including these are estimated. However, in this thesis, we will focus on the total ultimo and will not examine ultimo for specific features.

To be able use specific features in the Chain-Ladder we also have to extend our assumptions. The following extensions are introduced by Wüthrich [10] and is as follows

There exist parameters $f_0(\mathbf{x}), ..., f_{K-1}(\mathbf{x})$ such that for all $1 \le i \le I, 1 \le k \le K-1$ and $\mathbf{x} \in \mathcal{X}$

$$\mathrm{E}\left(C_{i,k+1}(\mathbf{x}) \,|\, C_{i,k}(\mathbf{x})\right) = f_k(\mathbf{x}) C_{i,k}(\mathbf{x}) + \mathrm{E}\left(C_{i,k+1}(\mathbf{x}) \,|\, C_{i,k}\right) I(C_{i,k}(\mathbf{x}) = 0)$$

There exist parameters $\sigma_0^2, ..., \sigma_{K-1}^2$ such that for all $1 \le i \le I, 1 \le k \le K-1$ and $\mathbf{x} \in \mathcal{X}$

$$\mathrm{Var}\left(C_{i,k+1}(\mathbf{x}) \,|\, C_{i,k}(\mathbf{x})\right) = \sigma_k^2 C_{i,k}(\mathbf{x}) + \mathrm{Var}\left(C_{i,k+1}(\mathbf{x}) \,|\, C_{i,k}\right) I(C_{i,k}(\mathbf{x}) = 0)$$

Where $I(C_{i,k}(\mathbf{x}) = 0)$ is an indicator which equals 1 when there is a zero-claim the year before. Further, we assume that $C_{ik}(\mathbf{x})$ of different accident years $1 \le i \le I$ or different features $\mathbf{x} \in \mathcal{X}$ are independent.

The reason why an extension of an intercept is made is because when working with sub-portfolios, there are situations where the cumulative value is zero for an accident year, given a feature value. There are two reasons this can occur. Either there have not been any claims reported for that accident year and feature value, or no payments have been made.

There are two possible ways to solve this issue. We can either reduce the dimensions of our feature vector or extend our assumptions as Wüthrich [10]. However, reducing the number of features removes the idea behind more homogeneous sub-groups. If all features are removed we are left with the regular Chain-Ladder. Hence we will use the approach with an intercept. This model can then be divided into two parts. One when $C_{ik} > 0$, which will be referred to as Part I, and one when $C_{ik} = 0$, Part II. We start by obtaining a model for Part I.

3.1 Part I : Claims with reported payments

As mentioned in Section 2, the first step is to choose a loss function. Given the assumptions, which are made in Section 1, it is reasonable to use the weighted square loss function. Given development year k, we can then express the loss function as

$$\begin{aligned} L_k &= \sum_{i=1}^{I-j} \sum_{\mathbf{x}:C_{ik}(\mathbf{x})>0} \frac{\big(C_{i,k+1}(\mathbf{x}) - C_{ik}(\mathbf{x})f_k(\mathbf{x})\big)^2}{\sigma_k^2 C_{ik}(\mathbf{x})} \\ &= \frac{1}{\sigma_k^2} \sum_{i=1}^{I-j} \sum_{\mathbf{x}:C_{ik}(\mathbf{x})>0} C_{ik}(\mathbf{x}) \left(\frac{C_{i,k+1}(\mathbf{x})}{C_{ik}(\mathbf{x})} - f_k(\mathbf{x}) \right)^2 \end{aligned}$$

where $f_k(\mathbf{x})$ will be our gradient boosted tree.

Once the loss function is selected, we proceed with finding the optimal tuning parameters for our model. LoB, injured part, claim code, and AQ are treated as categorical, while age will be treated as numerical. When using `gbm`-package in `R` the default for our tuning parameters are as follow; Interaction depth 1, i.e. no interaction, shrinkage 0.5, minimum observations in a node is 10 and bagging 0.5.

We start by examining the depth of each tree. The interaction depth in the `gbm`-package is stated as

$$\text{Interaction Depth} = \#\text{ of Terminal nodes} + 1.$$

If using the default depth, each tree is presented as a stump. However, Friedman and Hastie [4] suggest that the optimal number of terminal nodes, J, is $4 \leq J \leq 8$ with results relatively insensitive to particular choices in this range. To minimize the computational we will only examine interaction depths 1, 2, and 3, even thought this might not be optimal.

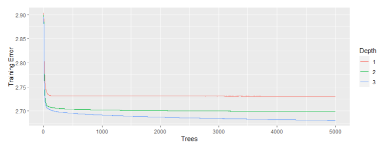
As mentioned in 2.2.2 the shrinkage factor, ν, is used to scale the contribution of each tree and is restricted to $0 < \nu \leq 1$. Where $\nu = 1$ imply no shrinkage. Here, Friedman [4], suggest that $\nu < 0.1$ appears to be the best strategy. Due to this, we will try both 0.1 and 0.01.

Bagging is used, as the other parameters, to prevent overfitting. The bagging fraction states how much training data should be used when fitting the model. Friedman and Hastie [4] suggest a bagging fraction of 0.5 or even lower for a higher number of observations. Since the last development quarters have scarce data, we will try 0.5 and 1.

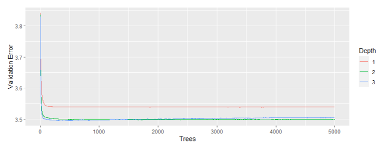
The last tuning parameter that should be tuned is minimum observations in a node. As mentioned above, there are few observations in the last development quarters, and therefore we will also investigate when allowing only one observation in the node.

A few things are worth mentioning before we continue analyzing our tuning parameters. To establish which parameter is optimal, we will evaluate using training and validation errors. The training data will consist of 90% of the data and the validation of 10%. Further, we will start by using the default values of the tuning parameters. Once we find the best, we will move on to the next. When the following parameter is analyzed, the optimal value of the previous is used. There might be the case that a change in a parameter would result in another optimal parameter, but that case will be disregarded here. The last thing worth mentioning is that we will use the same model for each development year, k. This is due to a lack of computational power and would take a long time. There might, however, be the case that each k has different optimal parameters.

The first tuning parameter that will be chosen is the interaction depth. What can be seen in Figure 4 (a), which presents the training error, a depth of 3 seems preferable. However, if we look at Figure 4 (b), we can see that there seems to be no difference between a depth of 2 or 3 when estimating previous unseen data. Since a less complicated model is preferred according to Occam's razor, we will proceed with a depth of 2.



(a)

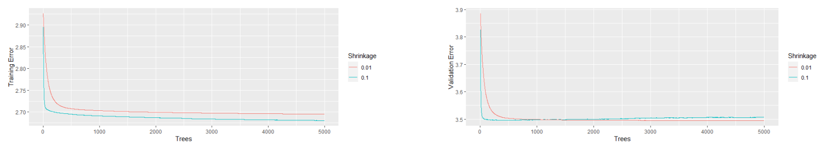


(b)

Figure 4: Training error (a) and validation error (b) for CY1 model with a interaction depth of 1,2 and 3. 90% of the data have been used for training and 10% for validation

We now move on the find the optimal value of shrinkage. As mentioned above, we will adjust the parameters as we proceed; hence we now find the optimal shrinkage given a depth of 2. Figure 5 present the training and validation error for shrinkage 0.1 and 0.01. As in the case of depth, there is a preferable value when looking at the training error. There is not as clear when looking at the validation error. When looking at the validation error, we can see that, as in the case of training error, shrinkage 0.1 is steeper in the beginning.

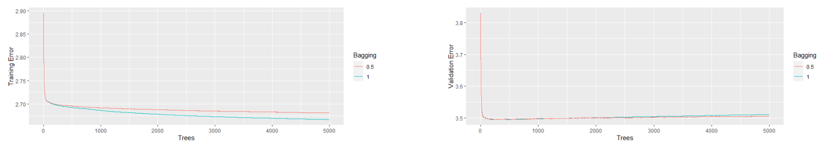
Another thing that can be noticed is that as the number of trees used in our model increases, the validation error for 0.1 exceeds the validation error for 0.01. A higher validation error means that it performs worse at unseen data. However, as we see the risk that the optimal number of trees will exceed 1000, we will use 0.1 shrinkage. A higher shrinkage often results in a lower number of trees used, hence, lowering the computational time.



(a) (b)

Figure 5: Training error (a) and validation error (b) for CY1 model with a shrinkage of 0.1 and 0.01. 90% of the data have been used for training and 10% for validation

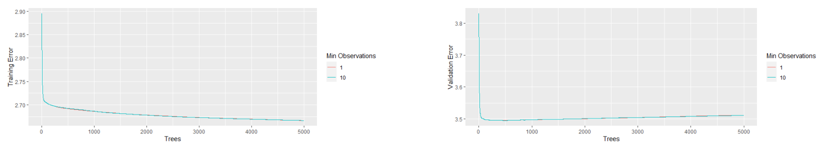
Figure 6 present the error for bagging. Here it is hard to draw any conclusions since the training and validation errors are similar. Hence we refer to Friedman and Hastie [4], which suggest a bagging value of 0.5.



(a) (b)

Figure 6: Training error (a) and validation error (b) for CY1 model with bagging of 0.5 and 1. 90% of the data have been used for training and 10% for validation

The last tuning parameter to be optimized is the minimum number of observations in a node. As can be seen in Figure 7 the errors are almost identical. Since both values result in the same error, a minimum number of observations will be set to 10. The value 10, instead of 1, is used since the prediction is done using the mean of the fitted values. If a minimum of 1 is used, there is a risk of high variance.



(a) (b)

Figure 7: Training error (a) and validation error (b) for CY1 model with minimum observation of 1 and 10. 90% of the data have been used for training and 10% for validation

To summarize, the tuning parameters used are Interaction Depth of 2, Shrinkage of 0.1, bagging set to 0.5, and the minimum number of observations in a node is 10. The last part of optimizing is the number of trees used in our model. To find this value, 5-fold cross-validation is used. The cross-validation for a different amount of trees is presented in Figure 8. As can be seen, the optimal number of trees is approximately 50-100. Hence, our prediction is confirmed that a higher number of trees than 1000 is not needed.

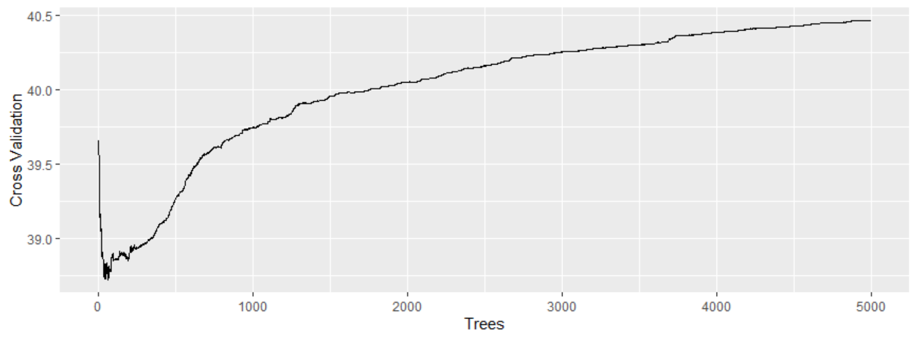


Figure 8: Cross validation for CY1 model where shrinkage equal 0.1, Interaction Depth of 2, bagging 0.5 and a minimum of 10 observations per node.

3.2 Part II : Claims without reported payments

The next step is to fit a model for the case when $C_{ik}(\mathbf{x}) = 0$. Here we will use the same approach as in Neural Network Applied to Chain-Ladder Reserving by Wüthrich, [10].

Given the assumptions that are made for Chain-Ladder, an accident year remains strictly positive once a claim has been reported. Due to this assumption, we only need to establish a model for the feature values that have no reported claims in the latest year, i.e., for all $C_{i,I-i}(\mathbf{x})$. To estimate models for each feature value equal to zero would result in a highly complex model. Hence, the following model will be defined for the lower triangle, $k + i > I$,

$$C^*_{ik} = \sum_{\mathbf{x}:\, C_{i,I-i}(\mathbf{x})=0} C_{ik}(\mathbf{x}).$$

Further, given this model, it is reasonable to assume the following approximation

$$C^*_{ik} \approx C_{i,I-i} \prod_{l=I-i}^{k-1} g_l^{(i)}$$

where $C_{i,I-i}$ is the total observed claim cost for accident year i and development year $I - i$. What can be noted is that the development factors are dependent on the accident year. This is reasonable since the same feature values might not be non-zero yearly.

When we have defined a model for the zero-claims we also need to estimate the devlopment factors. We start by defining the feature values for accident year $1 \leq k \leq i$ as

$$\mathcal{X}_k^{(i)} = \{\mathbf{x} \in \mathcal{X} \,:\, C_{k,I-i}(\mathbf{x}) = 0\}.$$

As mentioned above, all zero values are combined in a joint model. We can then express the cumulative value for development year $j > I - i$ and accident year k as

$$C^{(*i)}_{k,j} = \sum_{\mathbf{x} \in \mathcal{X}_k^{(i)}} C_{k,j}(\mathbf{x})$$

A reasonable way to estimate them is according to

$$\hat{g}^{(i)}_{I-i} = \frac{\sum_{k=1}^{i-1} C^{(*i)}_{k,I-i+1}}{\sum_{k=1}^{i-1} C_{k,I-i}}, \quad \text{and for } I-i < j < J \quad \hat{g}^{(i)}_j = \frac{\sum_{k=1}^{I-j-1} C^{(*i)}_{k,j+1}}{\sum_{k=1}^{I-j-1} C^{(*i)}_{k,j}}$$

The first development factor can be interpreted as how big part of the known value is usually observed the next year, given a set of features and the following development factors can be seen as usual Chain-Ladder estimations. The estimations of the development factors, for our data, are presented in Figure A.3.

3.3 Comparison

We have now calibrated our model which can be expressed as

$$\hat{C}_{iJ} = \sum_{\mathbf{x} \in \mathcal{X}} C_{i,I-i}(\mathbf{x}) \prod_{j=I-i}^{J-1} \hat{f}_j(\mathbf{x}) + C_{i,I-i} \prod_{j=I-i}^{J-1} \hat{g}^{(i)}_j$$

The model established in Part I estimates the IBNR with a Gradient Boosted Tree for each development year. Part II estimates the zero claims with the development factors given in Figure A.3. What can be seen in the figure is that the first development factors, the ones presented in the first diagonal, are small and decreases in earlier years. The zero-claim model assumes that the first unknown observation is a fraction of the last known value. It is reasonable to assume that the more rare claims are fewer; hence, the fraction is less than 1. Further, it is reasonable that the first development factor is smaller for earlier years since more data is known, resulting in fewer zero features.

The next step is to see how well the models perform. Since the data is generated using a simulator, we can compare the expected values with the true ones. The estimates for our model, the Chain-Ladder, and the true values are presented in Table 1. We have also included the MSEP for the Chain-Ladder. This is the prediction uncertainty related to one standard error.

What can be seen in the Table is that the GBM performs slightly better than the regular CL for the earlier years while the CL outperforms the latter years. Since our model has two parts, it could be either the non-zero model that performs slightly poorer, the GBM, or both. Since the latter years perform poorer, the earlier development factors must be the ones that do not perform as well. However, those development factors have the most information and should, however, give more stable estimates. It is, therefore, reasonable to start investigating the zero claim model.

	True Reserves	GBM Reserves	CL Reserves	$\sqrt{MSEP}$
1994	0	0	0	0
1995	1'536	1'529	1'510	63
1996	4'303	3'933	3'892	134
1997	7'937	6'888	6'860	276
1998	10'614	10'170	10'139	416
1999	15'044	15'144	14'807	598
2000	21'434	20'799	20'608	802
2001	32'006	30'789	30'416	1'173
2002	44'708	43'499	43'035	1'434
2003	62'155	65'582	64'796	2'136
2004	110'152	113'243	111'990	4'841
2005	239'630	253'857	235'675	10'745
Total	**549'519**	**565'433**	**543'728**	**12'789**

Table 1: True outstanding payments, GBM reserves, CL reserved and Mack's rooted mean square error prediction at time I. Values presented in 1'000

The true and the estimated reserves for our zero-features are presented in Table 2. As can be seen, the zero-features only explain approximately 1/3 deviation; hence, 2/3 of the deviation has its explanation by our GBM. To investigate this further, we investigate the average parameter. We use the same approach as in [10]. The average parameter is defined by

$$\overline{f}_{j-1}^{\text{GBM}} = \frac{\sum_{i=i}^{I-j} \sum_{\mathbf{x}:C_{i,j-1}(\mathbf{x})>0} \hat{f}_{j-1}(\mathbf{x}) C_{i,j-1}(\mathbf{x})}{\sum_{i=i}^{I-j} \sum_{\mathbf{x}:C_{i,j-1}(\mathbf{x})>0} C_{i,j-1}(\mathbf{x})}.$$

The marginalized version, i.e., the average CL factor considered as a function of the different labels, where one feature is fixed, is presented in Figure A.4 and A.5. Since the same simulator and parameters, except for the number of observations, as in [10] are used, we can compare the marginalized versions. To compare the factors, we first have to observe that each feature has the same distribution as in [10]. Figure A.2 shows the distribution of our features, which have the same as Wüthrich; hence we could compare our marginalized factors.

	True IBNR	Est. IBNR
1994	0	0
1995	1	9
1996	155	120
1997	105	174
1998	279	234
1999	254	404
2000	299	699
2001	859	963
2002	1'206	1'392
2003	1'009	2'124
2004	2'835	4'470
2005	58'129	63'100
Total	**65'131**	**73'689**

Table 2: True outstanding payments for zero-features and estimated using Wüthrich [10] at time I. Values presented in 1'000

What can be seen is that the marginalized versions look like Wüthrich's, except for the values of the estimated values. However, it is hard to conclude if the difference in the estimates is due to the different estimation approaches or because we are using fewer data points.

4 Comparing MSEP

As mentioned in Section 2.3 MSEP could be used to see how well a model predicts. To recall, MSEP is expressed as

$$\text{MSEP}(\hat{C}_{iI}) = \text{Var}(C_{iI}|D) + (\text{E}(C_{iI}|D) - \hat{C}_{iI})^2$$

where $\text{Var}(C_{iI}|D)$ is the the process variance and $(\text{E}(C_{iI}|D) - \hat{C}_{iI})^2$ is the estimation error.

Since MSEP is a measurement of uncertainty, extending the analysis and investigating whether the GBM is more or less uncertain than the original Chain-Ladder is convenient.

The data that was used previously, when estimating the ultimo, included zero feature claims. To compare the GBM approach with the Chain-Ladder, we create a new data set that does not have any zero features, i.e., all features have a claim in the first development year. Further, to reduce the computational time, we use fewer feature value combinations. When creating a new data set, the following feature values will be used

- LoB $\in$ {1,2,3,4}
- AQ $\in$ {1,2,3,4}
- Injured body part $\in$ {1,2,...,9}

Which gives us $|\mathcal{X}| = 4 \cdot 4 \cdot 9 = 144$ possible feature values.
To obtain the new data set, the following method has been used

1. Create an initial claims vector, $C_{i,1}$ $\forall \mathbf{x} \in \mathcal{X}$, $1994 \leq i \leq 2005$
2. Simulate the next development year for each feature value using

$$C_{i,k+1}(\mathbf{x}) = C_{ik}(\mathbf{x}) f_k(\mathbf{x}) + \sqrt{C_{ik}(\mathbf{x})}\sigma_k \epsilon$$

   where $\epsilon \sim N(0,1)$.

In this thesis we will use $\sigma_k = \sigma = 1$. The choice of $\sigma = 1$ might not be fully realistic given the assumptions made in the Chain-Ladder, and it could be expected that there is less variance in later development years than earlier. This data set, which consists of triangles for each feature value, will be referred to as "the initial data set."

When the data set has been obtained, we can calculate the MSEP for each model. We start with the MSEP for the Chain-Ladder, presented in Table 3. This can be calculated using the law of total conditional variance. For further information regarding the calculation, we refer to Mack's article regarding the calculation of the standard error [6]; however, since our sigma is known, we are able to use that instead of the estimated.

The calculation of the MSEP for the GBM can be divided into two parts. One part is the process variance, and the other is the estimation error.
The process variance for the GBM is calculated similarly to the Chain-Ladder. The difference is that the variance is calculated for each feature and then summed up. As the case in the Chain-Ladder, this is calculated using the law of total conditional variance.

The second part of the GBM MSEP is the estimation error. To estimate the estimation error, the following procedure is used

1. Create new triangles for each feature value $\in \mathcal{X}$ using the same $C_{i,1}(\mathbf{x}), \sigma$ and $f_k(\mathbf{x})$ as above.
2. Estimate the development factors using GBM on the new triangle.
3. Calculate the ultimo of the initial data set, using the new development factors.
4. Calculate the square of the difference between the initial data sets ultimo and the new ultimo.

This process is repeated N times, and the estimation error can then be estimated using the mean of the squared differences.

To estimate the estimation error, we have used N equal to 1000. The root square of the process variance, the estimation error, and the root square of the MSEP for the GBM are presented in Table 3.

	GBM			CL
AY	$\sqrt{\text{Estimation Error}}$	$\sqrt{Var}$	$\sqrt{MSEP}$	$\sqrt{MSEP}$
1994	0	0	0	0
1995	39	577	579	590
1996	56	824	826	824
1997	72	1 036	1 038	1 036
1998	73	1 195	1 197	1 195
1999	124	1 355	1 361	1 355
2000	165	1 486	1 495	1 485
2001	286	1 659	1 684	1 657
2002	319	1 788	1 817	1 786
2003	385	1 926	1 964	1 922
2004	602	2 092	2 177	2 087
2005	888	2 212	2 384	2 206

Table 3: Conditional root squared estimation error, process variance, and MSEP for GBM and Chain-Ladder for each accident year. Values presented in 1000s

What can be noted in the Table is that the Chain-Ladder MSEP is close to the process variance of the GBM. This might be due to too high development factors for our GBM. Higher development factors result in a higher estimation of the variance.

In Figure A.6 the ultimo's that have been used to calculate the estimation error are shown, together with the true ultimo. As can be seen, the GBM tends to overestimate the ultimo. One explanation is that the same parameters, as in the ultimo analysis, were used when the models were fitted. By looking at the histogram, it looks like those parameters were not correct.
It is hard to draw any conclusion from this analysis since the correct parameters were not chosen. An analysis to find the best-suited parameters should have been done to make a fair comparison.

5 Discussion

The scope of this thesis was to see if we could improve our Chain-Ladder reserve estimates by dividing the data into sub-portfolios and using machine learning to estimate our development factors. The approach that has been used is inspired by Wüthrich [10] with the modification that Gradient Boosted Trees have been used instead of Neural Networks.

What was observed was that the reserve estimates with GBM were in line with the Chain-Ladder for the earlier years. A benefit with GBM is that we are able to do a more granular analysis, since reserving is done by aggregating multiple feature estimations.

In the latter year, however, the GBM estimations were outperformed by the original Chain-Ladder. Either the feature space that is observed in the latter years differs from the previous years; hence the estimations of the development factors are poor. Another possible explanation is that the optimal tuning parameters have not been chosen. As mentioned earlier, the parameters have been chosen given the previous as fixed. By re-estimating the parameters, better predictions might be possible.

A few improvements should have been made if the time, knowledge, and computational power were available. First, we could not compare our results with Wüthrich's due to a lack of knowledge of `R` and lack of computational power. In our analysis, we were able to use 2'500'000 observations while Wüthrich [10] used 5'000'000. An increased number of observations would most likely improve our estimates but also be able to see the difference in our estimations. Another extension of Wüthrich's analysis was to divide the reserves into the different lines of businesses. Our model might have poorly performed on one line of business, hence an overall bad estimation.

The second thing is the choice of tuning parameter values. As mentioned earlier, we started by finding the optimal for one parameter, given the other. We then proceeded with this value and kept it fixed. A possible solution would be to return to the previous parameters and tune them once more until they converge. Further, since two parameters are continuous, shrinkage and bagging, multiple more values could be regarded than the few used in this thesis. However, since this is the case, there are infinite possible combinations; hence not possible to try them all.

Further, in this thesis, we used the same features as Wüthrich, i.e., all available. Since we used fewer observations than Wüthrich, there is a possibility that a more aggregated data set, e.g., not using claim code, would result in a better estimation. There is also a risk of overfitting when too many features are used. This could also be a parameter that changes over each claim year; hence this analysis should be done for each model.

As mentioned earlier, the division into sub-portfolios results in regular linear regression. Multiple machine learning techniques handle regression well, and the analysis could then be extended by adding more models.

6 References

[1] Gabrielli, A and Wüthrich, M. (2018). *An Individual Claims History Simulation Machine*. Risks, 6(2):29. https://doi.org/10.3390/risks6020029

[2] Gesmann, M., Murphy, D., Zhang, Y., Carrato, A., Wüthrich, M., Concina, F. and Dal Moro, E. (2022). *ChainLadder: Statistical Methods and Models for Claims Reserving in General Insurance*. R package version 0.2.15. https://CRAN.R-project.org/package=ChainLadder

[3] Greenwell, B., Boehmke, B., Cunningham, J., and GBM Developers. (2020). *gbm: Generalized Boosted Regression Models*. R package version 2.1.8. https://CRAN.R-project.org/package=gbm

[4] Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning*. New York, Ny Springer New York.

[5] James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An introduction to statistical learning : with applications in R*. Springer.

[6] Mack, T. (1993). *Distribution-free Calculation of the Standard Error of Chain Ladder Reserve Estimates*. ASTIN Bulletin, 23(2), 213-225. doi:10.2143/AST.23.2.2005092

[7] Mack, T. (1993). *Measuring the variability of Chain Ladder reserve estimates*, Munich Re

[8] Ridgeway, G. (2020) *Generalized Boosted Models: A guide to the gbm package*. https://cran.r-project.org/web/packages/gbm/vignettes/gbm.pdf

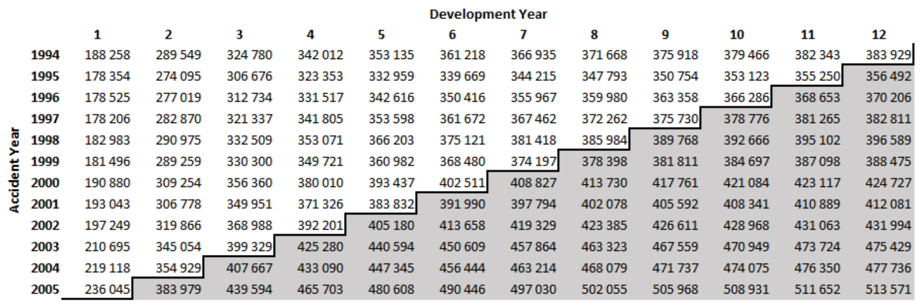
[9] Schnieper, R. (1991). *Separating True IBNR and IBNER Claims*. ASTIN Bulletin, 21(1), 111-127. doi:10.2143/AST.21.1.2005404

[10] Wüthrich, M. (2018). *Neural networks applied to Chain–Ladder reserving*. European Actuarial Journal, 8(2), 407-436. http://dx.doi.org/10.2139/ssrn.2966126

A Appendix

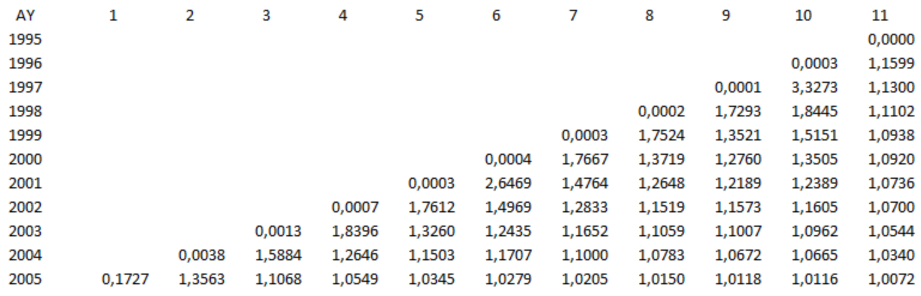
ClNr	LoB	cc	AY	AQ	age	inj_part
1	2	49	1994	2	32	71
2	2	46	1994	2	15	34
3	3	50	1994	1	39	14
4	2	41	1994	3	36	33
5	3	20	1994	1	59	53
6	1	6	1994	1	21	20
7	1	27	1994	1	46	36
8	4	46	1994	4	51	35
9	4	26	1994	4	36	21
10	2	40	1994	4	33	51

Table A.1: 10 first lines of feature data



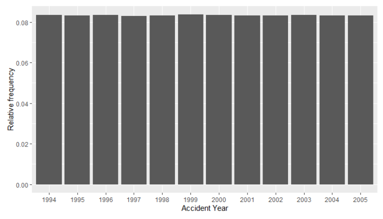
Accident Year	Development Year											
	1	2	3	4	5	6	7	8	9	10	11	12
1994	188 258	289 549	324 780	342 012	353 135	361 218	366 935	371 668	375 918	379 466	382 343	383 929
1995	178 354	274 095	306 676	323 353	332 959	339 669	344 215	347 793	350 754	353 123	355 250	356 492
1996	178 525	277 019	312 734	331 517	342 616	350 416	355 967	359 980	363 358	366 286	368 653	370 206
1997	178 206	282 870	321 337	341 805	353 598	361 672	367 462	372 262	375 730	378 776	381 265	382 811
1998	182 983	290 975	332 509	353 071	366 203	375 121	381 418	385 984	389 768	392 666	395 102	396 589
1999	181 496	289 259	330 300	349 721	360 982	368 480	374 197	378 398	381 811	384 697	387 098	388 475
2000	190 880	309 254	356 360	380 010	393 437	402 511	408 827	413 730	417 761	421 084	423 117	424 727
2001	193 043	306 778	349 951	371 326	383 832	391 990	397 794	402 078	405 592	408 341	410 889	412 081
2002	197 249	319 866	368 988	392 201	405 180	413 658	419 329	423 385	426 611	428 968	431 063	431 994
2003	210 695	345 054	399 329	425 280	440 594	450 609	457 864	463 323	467 559	470 949	473 724	475 429
2004	219 118	354 929	407 667	433 090	447 345	456 444	463 214	468 079	471 737	474 075	476 350	477 736
2005	236 045	383 979	439 594	465 703	480 608	490 446	497 030	502 055	505 968	508 931	511 652	513 571

Figure A.1: Cumulative values presented in 1000’s

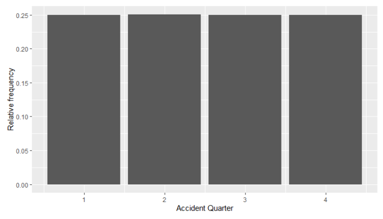


AY	1	2	3	4	5	6	7	8	9	10	11
1995											0,0000
1996										0,0003	1,1599
1997									0,0001	3,3273	1,1300
1998								0,0002	1,7293	1,8445	1,1102
1999							0,0003	1,7524	1,3521	1,5151	1,0938
2000						0,0004	1,7667	1,3719	1,2760	1,3505	1,0920
2001					0,0003	2,6469	1,4764	1,2648	1,2189	1,2389	1,0736
2002				0,0007	1,7612	1,4969	1,2833	1,1519	1,1573	1,1605	1,0700
2003			0,0013	1,8396	1,3260	1,2435	1,1652	1,1059	1,1007	1,0962	1,0544
2004		0,0038	1,5884	1,2646	1,1503	1,1707	1,1000	1,0783	1,0672	1,0665	1,0340
2005	0,1727	1,3563	1,1068	1,0549	1,0345	1,0279	1,0205	1,0150	1,0118	1,0116	1,0072

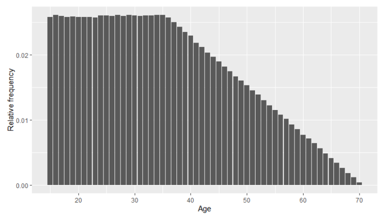
Figure A.3: Estimated zero-feature development factors.



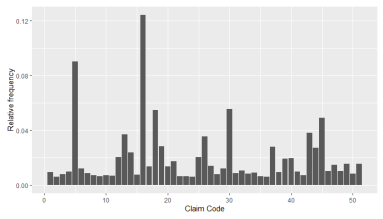
(a)



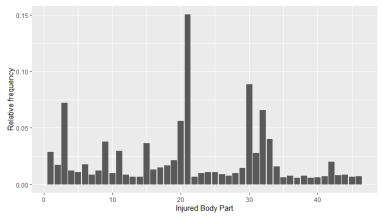
(b)



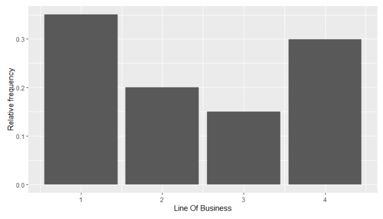
(c)



(d)



(e)



(f)

Figure A.2: Marginal distributions of the feature components Accident Year, Accident Quarter, Age, Claim Code, Injured Body Part and Line of Business.

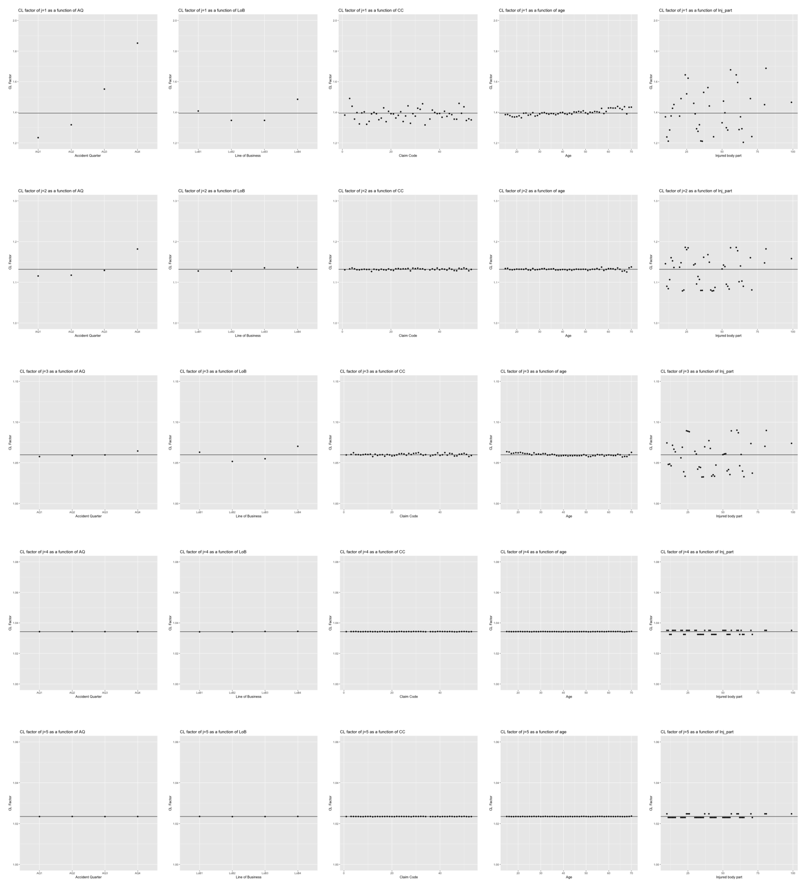


Figure A.4: Sensitives of the estimated CL factors $\hat{f}_{j-1}(\mathbf{x})$ in individual feature components for $j = 1, 2, .., 5$. Solid line shows the average development factor. Dots show the marginalized development factor given a feature value. Method taken from [10]

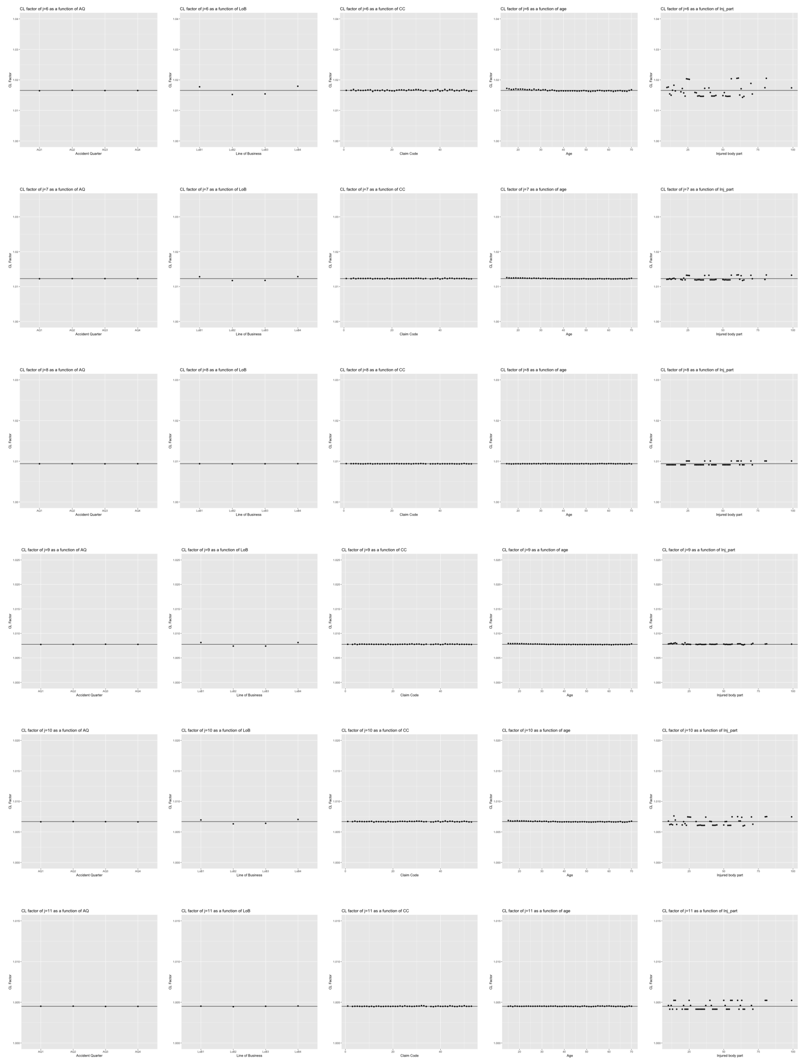


Figure A.5: Sensitives of the estimated CL factors $\hat{f}_{j-1}(\mathbf{x})$ in individual feature components for $j = 6, 2, .., 11$. Solid line shows the average development factor. Dots show the marginalized development factor given a feature value. Method taken from [10]

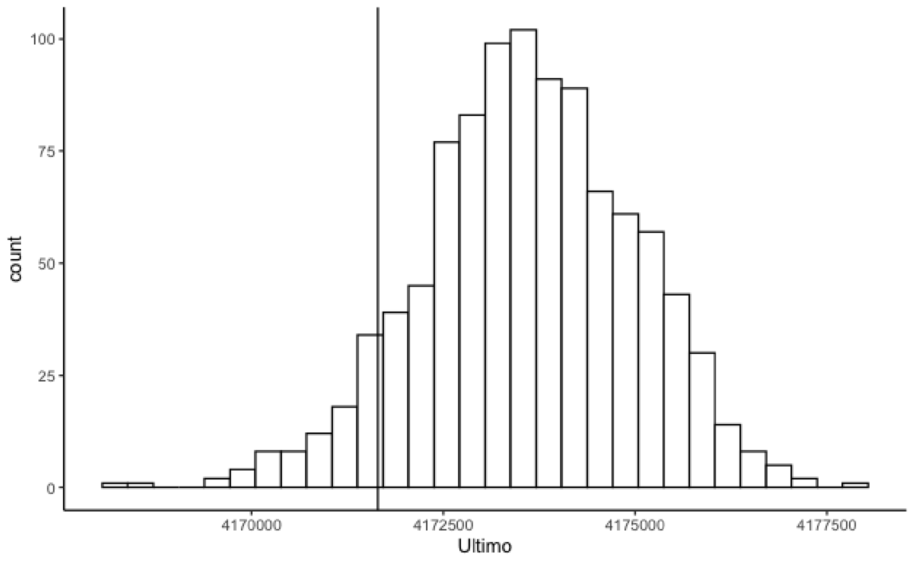


Figure A.6: Ultimo estimates used for calculating esimation error. The solid line represent the true ultimo.