Stockholms universitet

On Claims Reserving with Machine Learning Techniques

Vilma Härkönen

Masteruppsats i försäkringsmatematik
Master Thesis in Actuarial Mathematics

Masteruppsats 2021:4
Försäkringsmatematik
Juni 2021

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Stockholm University

Mathematical Statistics
Stockholm University
Master Thesis **2021:4**
http://www.math.su.se

On Claims Reserving with Machine Learning Techniques

Vilma Härkönen*

June 2021

Abstract

In this thesis we will explore the possibility of improving traditional non-life claims reserving models with machine learning techniques. We present three models; gradient boosting machines (GBM) and two neural network models. The first neural network reserving model is specified such that the claim counts and claim amounts are modelled separately in their own networks. The second one models the claim counts and amounts jointly in one network. The starting point for both neural network reserving models is the over-dispersed Poisson (ODP) reserving model estimates. Hence, the neural network models can be thought of as neural network boosting of the traditional ODP reserving model. We discuss the fitting procedure of the machine learning models and predict the outstanding reserves for simulated and real-world data. In addition, we calculate the mean squared error of prediction (MSEP) to examine the variation of the models. The presented models perform very well on the simulated data, especially GBM shows excellent performance. Moreover, the predictions are also improved for real-world data with GBM compared to the traditional Chain-Ladder reserving model.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: harkonen.vilma@gmail.com. Supervisor: Mathias Millberg Lindholm.

Acknowledgements

I would like to thank my supervisor Mathias Millberg Lindholm for his guidance and valuable feedback throughout this thesis project. I would also like to thank Daniel Andersson, Elin Roos and Folksam for interesting discussions and providing me data. Thank you to Henning Zakrisson for inspirational ideas. And finally, I would like to thank my fiancé Carlos for his support.

Contents

1 Introduction

Actuarial science focuses on quantifying the risk of an event occurring in the future. A key question for actuaries is to estimate the future claim payments so that the insurance company is able to cover all future claims. Normally, this problem is solved by finding some appropriate regression function to estimate the average claim costs. In this thesis we explore the possibilities to improve claims reserving using machine learning techniques. We use the traditional over-dispersed Poisson (ODP) model as benchmark. The machine learning techniques used in this thesis are neural networks and gradient boosting machines (GBM). The models are applied to simulated Swiss claims data from [6] and we test different types of fitting and tuning to find the optimal calibration. In addition, we calculate predicted outstanding reserves and compute the mean squared error of prediction (MSEP). Thereafter we apply the models to rather more complicated real-world data that is from a Swedish insurance company Folksam.

Today, there are several machine learning models but these are more seldom used in the claims reserving context. In [15] a blended model that combines a more conventional regression model with a neural network model is introduced. The idea is to use an appropriate regression model to obtain the initial parameter estimates that is then used in a neural network as inputs. Hence, the goal is to calibrate the estimation of the outcome from the regression model with a more sophisticated machine learning model. This can also be thought of as neural network boosting of the traditional reserving model.

In [5] the authors present a neural network model to model the claim amounts where they calibrate the estimation from the ODP model. Another approach is presented in [4] where the authors implement a double feed-forward network to model the claim counts and claim amounts simultaneously. The results from both of these articles indicate that the traditional reserving model has potential for improvement as the predicted outstanding reserves are considerably better with neural network boosting.

However, in the real-world industry the outstanding reserves are often divided into RBNS (reported but not settled) and IBNR (incurred but not reported) reserves. The authors in [10] take reporting delay after the claim date into account which allows for separate modelling of the RBNS and IBNR reserves. Yet, the modelling is still performed on an aggregated level with the neural network model from [5]. A tree-based method, GBM, is also introduced by the authors into the claims reserving context which shows great performance. In addition, an advantage of GBM is its simple implementation with e.g. the statistical software `R`.

Tree-based models can be divided into regression trees and classification trees but here we focus on the first type. The idea of tree-based models is to successively divide the feature space into rectangles. Moreover, the regression tree models are easy to illustrate and interpret which have contributed to their popularity. In previous literature tree-based models have been applied to individual claims reserving, see [2] and [16] where claim counts are modelled with regression trees. These papers have found tree-based models promising

in the claims reserving context.

1.1 Outline

In Section 2 we present the theoretical framework of the models used in this thesis. Thereafter the models are evaluated on simulated data in Section 3. Moreover, we apply the models to real-world data in Section 4. Lastly, we discuss the results in Section 5.

2 Methods

In this section we present the theoretical framework that is applied in this thesis. First in Section 2.1 we will discuss over-dispersed Poisson reserving models. Then we move on to the machine learning models and present GBM in Section 2.2. Neural networks are discussed in Section 2.3. Finally, we will cover conditional mean squared error of prediction in Section 2.4.

2.1 Over-Dispersed Poisson Reserving Models

For the general theory used in this section we refer to [13] and for the more specific reserve modelling application used in this thesis to [10].

The over-dispersed Poisson distribution differs from the Poisson distribution by allowing the variance to be proportional to the mean, i.e. not equal as for a Poisson distribution. Let X_{ij} be an incremental claim payment made $j \in \{0, ..., I-1\}$ years later for claims occurred during accident year $i \in \{1, ..., I\}$. Further, the X_{ij}'s are independent over-dispersed Poisson distributed and have the following mean and variance

$$\mathbb{E}[X_{ij}] = \mu_{ij} \quad \text{and} \quad \text{Var}(X_{ij}) = \phi\mu_{ij}, \tag{1}$$

where $\phi > 0$ is the over-dispersion parameter. With log-link function the mean can be expressed as

$$\mu_{ij} = \exp\{c + \alpha_i + \beta_j\} \tag{2}$$

where c, α_i and β_j are the covariate coefficients for the intercept, accident year i and development year j. When used in practice, all parameters are replaced by the maximum likelihood (ML) estimates $\hat{\mu}_{ij}$, $\hat{c}$, $\hat{\alpha}_i$ and $\hat{\beta}_j$. It should be noted that the over-dispersion parameter does not affect the ML parameter estimation, instead it increases the standard deviation.

One stochastic model producing the Chain Ladder model is the over-dispersed Poisson model as in (1) with link function

$$\log(\mu_{ij}) = \log(e_i) + c + \alpha_i + \beta_j \Leftrightarrow \mu_{ij} = e_i \exp\{c + \alpha_i + \beta_j\} \tag{3}$$

where the e_i's are exposures that are modelled as offsets in the GLM framework and by default set to 1. In order to avoid over-parametrisation it is important to set the following constraint $\alpha_1 = \beta_1 = 0$. The model parameters are estimated on the upper left triangle of data, i.e. the X_{ij}'s where $i + j \leq I$, and the target values, the X_{ij}'s where $i + j > I$, are predicted according to $\hat{\mu}_{ij} = e_i \exp\{\hat{c} + \hat{\alpha}_i + \hat{\beta}_j\}$, see [13].

In this thesis we will mainly work with more granular data where we have divided the development dynamics into reporting delay $j \in \{0, ..., I-1\}$ and payment delay $k \in \{0, ..., d\}$ after reporting the claim, see [10]. In particular, we let $X_{ijk}|N_{ij} \sim \text{ODP}(\mu^X_{ijk} N_{ij}, \phi^X)$ with mean

$$\mathbb{E}[X_{ijk}|N_{ij}] = \mu^X_{ijk} N_{ij} = N_{ij} \exp\{c^X + \alpha^X_i + \beta^X_j + \gamma^X_k\} \tag{4}$$

and variance

$$\text{Var}[X_{ijk}|N_{ij}] = \phi^X \mu^X_{ijk} N_{ij} = \phi^X \mathbb{E}[X_{ijk}|N_{ij}] \tag{5}$$

where β_j and γ_k are now the covariate coefficients for reporting delay j and payment delay k. The exposures, the e_i's in (3), are replaced with the observed number of claims N_{ij}'s per accident year i and reporting delay j.

We can analogously define the over-dispersed Poisson model for the claim counts N_{ij}'s that depend only on the accident year i and reporting delay j by $N_{ij} \sim \text{ODP}(\mu^N_{ij}, \phi^N)$

$$\mathbb{E}[N_{ij}] = \mu^N_{ij} = \exp\{c^N + \alpha^N_i + \beta^N_j\} \tag{6}$$

and

$$\text{Var}[N_{ij}] = \phi^N \mu^N_{ij} = \phi^N \mathbb{E}[N_{ij}] \tag{7}$$

Hence the outstanding reserve R_i for accident year i is defined as

$$R_i := R^{\mathcal{R}}_i + R^{\mathcal{I}}_i \tag{8}$$

where $R^{\mathcal{R}}_i$ and $R^{\mathcal{I}}_i$ denote the outstanding RBNS and IBNR reserves. If we take the expectation of these reserves we obtain

$$\mathbb{E}[R^{\mathcal{R}}_i|\mathcal{N}_0] = \sum_{j=I-i+1}^{I-1+d} \sum_{k=j-I-1}^{j\wedge d} \mu^X_{i,j-k,k} N_{i,j-k} \tag{9}$$

and

$$\mathbb{E}[R^{\mathcal{I}}_i|\mathcal{N}_0] = \sum_{j=I-i+1}^{I-1+d} \sum_{k=0\vee(j-I+1)}^{(j-I+i-1)\wedge d} \mu^X_{i,j-k,k} \mu^N_{i,j-k}. \tag{10}$$

where $\mathcal{N}_0 := \sigma\{N_{ij} : i + j \leq I, i = 1, ..., I, j = 0, ..., I-1\}$ is the known number of claims at time $i + j \leq I$, see [10].

2.2 Tree-Based Gradient Boosting Machines

The idea of the tree-based models is to divide the feature space into several smaller rectangles and then fit a simple model, e.g. a constant, in each of these smaller areas. The whole feature space can thereafter be plotted in a tree-like plot based on the partitioning of the feature space. Tree models are popular due to their easy interpretation and illustration. Tree models can generally be dived into regression trees and classification trees, but here we will focus on the first one, see [8]. The main reference in this section for the general theory is [8] and for the application to claims reserving we refer to [10].

2.2.1 Regression Trees

In order to grow a regression tree, the algorithm needs to automatically decide which features should be partitioned and where to make the split. Assume that we have p input parameters and the data is in the form $(\boldsymbol{x}_i, y_i)$ for $i = 1, ..., n$ where y_i is the response and $\boldsymbol{x}_i = (x_{i,1}, ..., x_{i,p})$ are the features for observation i. If we decide to partition the feature space into A regions $R_1, ..., R_A$ and we model the response as a constant θ_a in each region then we would have:

$$f(\boldsymbol{x}) = \sum_{a=1}^{A} \theta_a I(\boldsymbol{x} \in R_a) \tag{11}$$

In case the minimisation criterion is the sum of squares $\sum(y_i - f(\boldsymbol{x}_i))^2$ then the best $\hat{\theta}_a$ is the average of y_i in region R_a:

$$\hat{\theta}_a = \text{ave}(y_i | \boldsymbol{x}_i \in R_a). \tag{12}$$

The minimisation problem above is computationally difficult and therefore in order to make the minimisation problem possible, we construct an approximate greedy algorithm. This means that we only optimize the regions at a given depth, leaving the previously optimised regions as fixed. Then the algorithm continues to the next level and so on. Hence, the same covariates may appear on different levels in the same tree. Consider starting with all of the data with splitting variable q and split point s, we define a pair of half-planes

$$R_1(q, s) = \{\boldsymbol{x} | x_q \leq s\} \quad \text{and} \quad R_2(q, s) = \{\boldsymbol{x} | x_q > s\} \tag{13}$$

The goal is to find a splitting variable v and a split point s that solve

$$\min_{q,s}[\min_{\theta_1} \sum_{\boldsymbol{x}_i \in R_1(q,s)} (y_i - \theta_1)^2 + \min_{\theta_2} \sum_{\boldsymbol{x}_i \in R_2(q,s)} (y_i - \theta_2)^2] \tag{14}$$

The optimisation problem in (14) is solved for any q and s by

$$\hat{\theta}_1 = \text{ave}(y_i | \boldsymbol{x}_i \in R_1(q, s)) \quad \text{and} \quad \hat{\theta}_2 = \text{ave}(y_i | \boldsymbol{x}_i \in R_2(q, s)) \tag{15}$$

where ave$(y_i|\boldsymbol{x}_i \in R_1(q,s))$ corresponds to the average of y_i in region R_1, see [8].

Like any other regression model, a binary regression tree can be thought of as producing the expected value of Y given $\boldsymbol{x}$. Since in our case we focus on ODP models, we can define

$$T(\boldsymbol{x};\Theta) := u(\mathbb{E}[Y|\boldsymbol{x};\Theta]) \tag{16}$$

where $u(\cdot)$ is a given link-function, $\boldsymbol{x}$ is the feature vector $\boldsymbol{x} = (x_i, ..., x_p)$ and $\boldsymbol{x}_i = (x_{i,1}, ..., x_{i,p})$ and Θ is the parameter vector. The distribution of $Y|\boldsymbol{x}$ implies a given loss function to be used in the estimation of $\boldsymbol{x}$. Since we are interested in ODP models a Poisson likelihood, or deviance, is an appropriate loss function. A binary regression tree of depth v splits the feature space into 2^v regions R_a where $a = 1, ..., 2^v = A$ and we can re-write (11) as

$$T(\boldsymbol{x};\Theta) = \sum_{a=1}^{2^v} \theta_a I(\boldsymbol{x} \in R_a) \tag{17}$$

where Θ is the feature vector containing all the θ_a's and additional parameters needed to define the R_a's. We can also define (12) more formally as

$$\theta_a := \arg\min_{\theta} - \sum_{i:\boldsymbol{x}_i \in R_a} L(y_i, T(\boldsymbol{x}_i;\Theta)) \tag{18}$$

where

$$T(\boldsymbol{x}_i;\Theta) = \theta, \quad i : \boldsymbol{x}_i \in R_a \tag{19}$$

and where L denotes the Poisson log-likelihood function. In tree based modelling the regions R_a are often referred to as the "leaves" of the tree and v denotes the depth of the tree i.e. how many splits there will be in the feature space, see [10].

While growing a tree a question may arise; how big tree should we grow? The problem with a large tree is that it may overfit data and a too small tree might not capture the structure of the data well enough. The procedure above will always grow a tree of depth v but this might not always be the optimal choice for the depth. The optimal choice of the tree size must always be adapted to data. A common method in choosing a tree size is called pruning where the algorithm stops when some minimum node size is reached, see [8]. For example we could decide to have at least 10 observations in each leaf.

Moreover, an advantage of regression trees is that they handle several features easily. The problem though is the risk of overfitting which can be handled with regularisation, for example boosting, see [8]

2.2.2 Gradient Tree-Boosting

The idea of boosting is to combine the output of many "weak" predictors to construct a more powerful predictor. This is done by sequentially applying the weak prediction

algorithm to repeatedly modified versions of the data resulting in several weak predictors. The final prediction will then be a weighted sum of the weaker predictions.

The gradient g_{iu} is defined as

$$g_{iu} = \left[\frac{\partial L(y_i, G(\boldsymbol{x}_i))}{\partial G(\boldsymbol{x}_i)}\right]_{G(\boldsymbol{x}_i)=G_{u-1}(\boldsymbol{x}_i)}. \tag{20}$$

The aim of gradient boosting is to find a tree with low depth such that the predictions at the uth iteration are as close as possible to the point-wise negative gradient of the loss function

$$\tilde{\Theta}_u = \arg\min_{\Theta} \sum_{i=1}^{n} (-g_{iu} - T(\boldsymbol{x}_i; \Theta))^2, \tag{21}$$

where squared error is used to measure the closeness and $\tilde{\Theta}_u = \{R_{vu}, \theta_{vu}\}_1^{V_u}$. Hence the tree is fitted to the negative gradient values. The generic algorithm for gradient descent tree-boosting regression is as follows

Algorithm 1: The gradient tree-boosting algorithm

1. Initialise $G_0(\boldsymbol{x}) = \arg\min_\theta \sum_{i=1}^{n} L(y_i, \theta)$
2. For u=1 to U:
   (a) For $i = 1, ..., n$ compute
   $$-g_{iu} = -\left[\frac{\partial L(y_i, G(\boldsymbol{x}_i))}{\partial G(\boldsymbol{x}_i)}\right]_{G(\boldsymbol{x}_i)=G_{u-1}(\boldsymbol{x}_i)}.$$
   (b) Fit a regression tree to the targets g_{iu} giving terminal regions $R_{vu},\ v = 1, ..., V_u$.
   (c) For $v = 1, ..., V_u$ compute
   $$\theta_{vu} = \arg\min_\theta \sum_{\boldsymbol{x}_i \in R_{vu}} L(y_i, G_{u-1}(\boldsymbol{x}_i) + \theta)$$
   (d) Update $G_u(\boldsymbol{x}) = G_{u-1}(\boldsymbol{x}) + \sum_{v=1}^{V_u} \theta_{vu} I(\boldsymbol{x} \in R_{vu})$
3. Output $\hat{G}(\boldsymbol{x}) = G_U(\boldsymbol{x})$.

The tuning parameters U and $V_u, u = 1, ..., U$ corresponds to the number of iterations and the sizes of each tree respectively. In case of too large U we run a risk of overfitting and the choice of U should always be adapted to data. In order to avoid overfitting we can also add other regularisation strategies to the tree model such as shrinkage and bagging. By introducing a shrinkage parameter ξ, that takes values between 0 and 1, we scale the contribution of each tree to the current gradient approximation. Hence, line 2.(d) in Algorithm 1 would be replaced with

$$G_u(\boldsymbol{x}) = G_{u-1}(\boldsymbol{x}) + \xi \sum_{v=1}^{V_u} \theta_{vu} I(\boldsymbol{x} \in R_{vu}) \tag{22}$$

and after U iterations we would have

$$G_U(\boldsymbol{x}) = \xi \sum_{v=1}^{V_U} \theta_{vU} I(\boldsymbol{x} \in R_{vU}). \tag{23}$$

That is, the gradient boosting machine predictor $f^{GBM}(x;\hat{\theta})$ is given by

$$f^{GBM}(\boldsymbol{x};\hat{\Theta}) := \xi \sum_{v=1}^{V_U} T(\boldsymbol{x};\hat{\Theta}). \tag{24}$$

Bagging is a bootstrap averaging strategy to improve the performance of the predictor where only a subset of the observations are used in each updating of $G_u(\boldsymbol{x})$. A common value for the bagging parameter η is $\frac{1}{2}$ meaning that at each iteration we sample without replacement half of the training observations. The advantage of bagging is that it reduces the computation time by fraction η and can reduce the variance of the predictor, see [8].

Since we are using a Poisson model with log-link function the estimated reserving model will be

$$\begin{cases} \hat{\mu}^X := \hat{\mathbb{E}}[X_{i,j,k}|N_{i,j}] = N_{i,j} \exp\{f^{\text{GBM},X}(i,j,k;\hat{\Theta})\} \\ \hat{\mu}^N := \hat{\mathbb{E}}[N_{i,j}] = \exp\{f^{\text{GBM},N}(i,j;\hat{\Theta})\} \end{cases} \tag{25}$$

and the estimates of the over-dispersion parameters $\hat{\phi}^X$ and $\hat{\phi}^N$ are based on Pearson or deviance residuals.

We emphasise that Algorithm 1 is a generic gradient descent tree-boosting algorithm. For implementation of GBM we use `gbm`-package in `R` and refer to the documentation of the package, `utils::browseVignettes("gbm")`, for more details.

2.2.3 Loss Function

The loss function that is minimised in the GBM reserving models in (25) is the unscaled Poisson deviance. In [12] this is defined as

$$L(y_i;\mu_i) = 2\sum_i \{y_i \log(y_i/\mu_i) - (y_i - \mu_i)\}. \tag{26}$$

By changing the notation of (26) as in [5] we obtain the loss function for the claim counts function

$$L^N(\mu^N_{i,j}, \mathcal{D}^N_I) = 2 \sum_{i+j\le I} \mu^N_{i,j} - N_{i,j} + N_{i,j} \log\left(\frac{N_{i,j}}{\mu^N_{i,j}}\right), \tag{27}$$

where $\mathcal{D}^N_I$ denotes the upper triangle defined as $\mathcal{D}^N_I = \{N_{i,j}, i = 1,...,i, j = 0,...,J, i+j \le I\}$. For the claim amounts we minimise the following

$$L^X(\mu^X_{i,j,k}, \mathcal{D}^X_I) = 2 \sum_{i+j+k\le I} \mu^X_{i,j,k} - X_{i,j,k} + X_{i,j,k} \log\left(\frac{X_{i,j,k}}{\mu^X_{i,j,k}}\right), \tag{28}$$

where $\mathcal{D}^X_I = \{X_{i,j,k}, i = 1,...,i, j = 0,...,J, k = 0,...,K, i+j+k \le I\}$.

2.3 Neural Networks

In this section we will discuss two neural network models that we call simple and double neural networks. The first one models the claim amounts and the claim counts separately in their own networks, see Section 2.3.3. The second one, presented in Section 2.3.4, allows joint modelling of the claim counts and the claim amounts in a two feed-forward network. But first, we introduce some general background used in the network in Sections 2.3.1-2.3.2. We use mainly [7] for the general theory of neural networks and follow [5] and [4] closely for the theory of the simple and neural network reserving models, respectively.

2.3.1 Feed-Forward Neural Network

A feed-forward neural network is an algorithm that is inspired by neuroscience. It often consists of several different functions, $\mathbf{z}^{(h)}$'s, where $h = 1, ..., H+1$. The information flows from the input vector $\mathbf{z}^{(0)}$ to the intermediate functions $\mathbf{z}^{(1)}, ..., \mathbf{z}^{(H)}$ and finally to the output $\mathbf{z}^{(H+1)}$, hence the name feed-forward network.

For example a three layer feed-forward network is given by $\mathbf{z}(\boldsymbol{x}) = \mathbf{z}^{(3)}(\mathbf{z}^{(2)}(\mathbf{z}^{(1)}(\boldsymbol{x})))$ where $\boldsymbol{x}$ is a vector of covariates and $\mathbf{z}^{(1)}$ is the first layer, $\mathbf{z}^{(2)}$ is the second layer and so on. The layers, $\mathbf{z}^{(h)}$'s, are defined as

$$\mathbf{z}^{(h)} = f^{(h)}(\mathbf{b}_h + \langle \mathbf{w}_h, \mathbf{z}^{(h-1)}(\boldsymbol{x}) \rangle) \tag{29}$$

where $\mathbf{z}^{(h-1)} \in \mathbb{R}^{q_{h-1}}$, $\mathbf{b}_h \in \mathbb{R}^{q_h}$ are the biases, $\mathbf{w}_h \in \mathbb{R}^{q_{h-1} \times q_h}$ the weights and $f^{(h)}$ some function that is also called as the activation function which is applied element-wise. In case of exponential activation we have $f^{(h)} = \exp\{\cdot\}$. The operation $\langle \cdot, \cdot \rangle$ denotes the scalar product in Euclidean space. The depth of the model is given by the length of the function chain. In the example above, the depth of the network $\mathbf{z}(\boldsymbol{x})$ is 3.

The goal of the feed-forward neural network is to provide as close approximation of the outcome, $\mathbf{z}$, as possible. The other layers $\mathbf{z}^{(h)}$ where $h = 1, ..., H$ are called as the hidden layers since we do not obtain the outcome of these layers. Here we have two hidden layers. The hidden layers have q_h units that can be thought as vector-to-scalar functions since they take in a vector and give an output as a scalar. For more on feed-forward networks, see [7].

2.3.2 Regularisation

Neural networks have a tendency of easily overfitting data as these are highly flexible models. The number of parameters increases for each layer and neuron. In order to avoid this there are several different regularisation methods. A common regularisation method is L2-regularisation, also known as the ridge regression or weight decay in a neural network context. This method penalises the fit of the model by adding a term $\lambda \sum_{j=1}^{p} w_j^2$ to the objective function that it minimised, where p is the total number of parameters in the

model and λ is the shrinkage factor. The effect of the shrinkage is larger for higher values of λ. The residual sum of squares with L2-regularisation becomes

$$\tilde{\mathbf{w}} = \arg\min_{w}\{\sum_{i=1}^{n}(y_i - w_0 - \sum_{j=1}^{p} x_{ij}w_j)^2 + \lambda\sum_{j=1}^{p} w_j^2\}, \tag{30}$$

where $i = 1, ..., n$ denotes the number of observations. The idea of L2-regularisation is to penalise the objective function for large weights and thus, prevent the weights from exploding, see [8].

Another regularisation method is so-called dropout which can be seen as a stochastic regularisation method. The idea of dropout is to randomly drop hidden units from the network during training. By dropping neurons we mean temporarily remove random units from the network. This corresponds to sampling a "thinned" network during training. Let $r_j^{(h)} \sim$ Bernoulli$(1-\rho)$, where $\rho \in \{0, 1\}$ is called the dropout rate. Then the "thinned" outputs from a hidden layer $h-1$ are defined as

$$\tilde{\mathbf{z}}^{(h-1)} = \mathbf{z}^{h-1} \odot \mathbf{r}^{(h-1)}, \tag{31}$$

where $\odot$ denotes the Hadamard product. Hence, (29) becomes

$$\mathbf{z}^{(h)} = f^{(h)}(\mathbf{b}_h + \langle \mathbf{w}_h, \tilde{\mathbf{z}}^{(h-1)} \rangle). \tag{32}$$

The motivation for dropout is that by dropping some of the neurons, the neurons have to learn to co-operate with a random sample of other neurons and thus create useful features on its own and not rely on the other neurons to correct its mistakes, see [14].

2.3.3 A Neural Network Reserving Model

In this section we will present the neural network reserving model where the claim counts and claim payments are modelled separately. We call this model a simple neural network model. All references in this section are with respect to [5] if not stated otherwise and throughout this section we use the notation $Y \in \{N, X\}$ where N denotes the number of claims and X the claim payments.

Figure 1 illustrates the architecture of the simple neural network model. The first part of the model is the blue area that is the input variables and the embedding layers. Notice that for the claim amount network model we also have a third input variable depending on k. The black boxes illustrate the hidden layers in the neural network structure and the red circle is the output. The green line is called skip-connection that directly connects the ODP reserving model structure to the neural network structure.

2.3.3.1 Embedding Layers

A crucial point of the neural network reserving model is that the ODP reserving model can easily be integrated to a more complex network modelling. The ODP reserving model

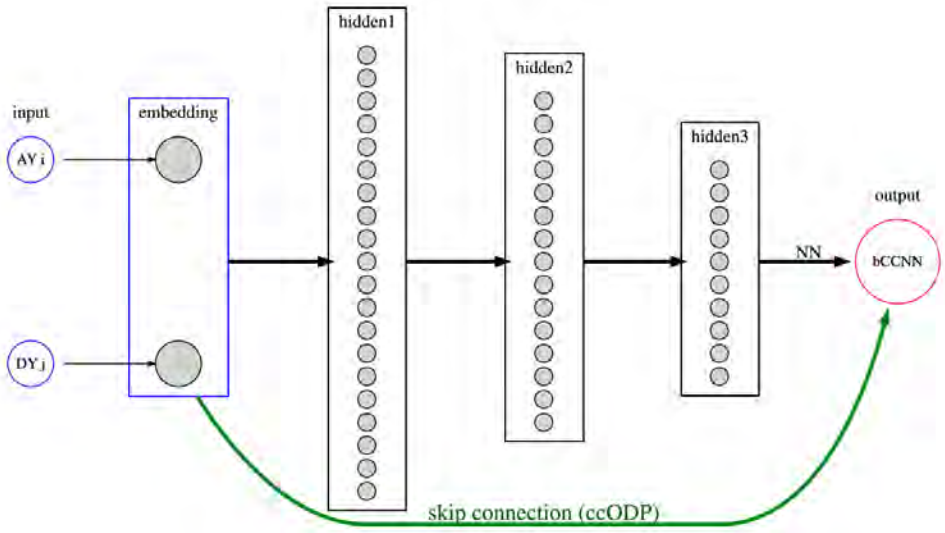

Figure 1: Illustration of the neural network reserving model architecture.
Source: Figure 1 in [5].

implicitly assumes a categorical representation of the covariates accident year i, reporting delay j and payment delay k as each year gets its own parameter coefficient. By adopting embedding layers we are able to blend the ODP reserving model with a neural network structure.

Embedding layers are used when categorical data is applied to a neural network. The idea of embedding layers is to learn a low-dimensional representation of the categorical variable by mapping every level of the categorical variable to a vector in $\mathbb{R}^p$ for some $p \in \mathbb{N}$. The embedding layers are then used as variables that can be trained in the neural network.

Since the ML-estimates from the ODP reserving model are already reasonable representations of the input variables accident year i, reporting delay j, and payment delay k, we choose these as mappings for the one-dimensional embedding layers

$$\boldsymbol{\alpha}(\cdot) : (1, ..., I) \ \to \mathbb{R}, \quad i \to \hat{\boldsymbol{\alpha}}(i) = \hat{\alpha}_i^Y \tag{33}$$

$$\boldsymbol{\beta}(\cdot) : (1, ..., J) \ \to \mathbb{R}, \quad j \to \hat{\boldsymbol{\beta}}(j) = \hat{\beta}_j^Y \tag{34}$$

and

$$\boldsymbol{\gamma}(\cdot) : (1, ..., K) \ \to \mathbb{R}, \quad k \to \hat{\boldsymbol{\gamma}}(k) = \hat{\gamma}_k^X \tag{35}$$

where $\boldsymbol{\alpha}$ denotes the covariate coefficients for accident year, $\boldsymbol{\beta}$ for reporting delay, $\boldsymbol{\gamma}$ for payment delay.

2.3.3.2 Cross-Classified Structure

The embedded input parameters are then adopted in an exponential activation function. Hence we have the following cross-classified neural network structure for the claim counts part

$$(i,j) \to \mu^{CC,N}(i,j) = \exp\{\hat{\boldsymbol{\alpha}}^N(i) + \hat{\boldsymbol{\beta}}^N(j)\}. \tag{36}$$

For the claim amounts part we define the following

$$(i,j,k) \to \mu^{CC,X}(i,j,k) = \exp\{\hat{\boldsymbol{\alpha}}^X(i) + \hat{\boldsymbol{\beta}}^X(j) + \hat{\boldsymbol{\gamma}}^X(k)\} \tag{37}$$

Thus, equations (36) and (37) define the neural network models of the classical ODP reserving model. The aim is to connect this cross-classified structure to a more general neural network structure in order to calibrate the predictions from the ODP reserving model, see green line in Figure 1.

2.3.3.3 Neural Network Structure

Next we define the neural network structure of the model, see black boxes in Figure 1. The embedding layers defined in (33) - (35) are used as inputs in the first hidden layer $\mathbf{z}^{(0),Y}$. For the claim counts part that is

$$\mathbf{z}^{(0),N} = \mathbf{z}^{(0)}(i,j) = (\hat{\boldsymbol{\alpha}}(i), \hat{\boldsymbol{\beta}}(j)) = (\hat{\alpha}_i^N, \hat{\beta}_j^N) \in \mathbb{R}^{q_0}, \tag{38}$$

where $q_0 \coloneqq q_0^N$ denotes the number of neurons in the first hidden layer. For the claim amounts part we have

$$\mathbf{z}^{(0),X} = \mathbf{z}^{(0)}(i,j,k) = (\hat{\boldsymbol{\alpha}}(i), \hat{\boldsymbol{\beta}}(j), \hat{\boldsymbol{\gamma}}(k)) = (\hat{\alpha}_i^X, \hat{\beta}_j^X, \hat{\gamma}_k^X) \in \mathbb{R}^{q_0}. \tag{39}$$

where $q_0 \coloneqq q_0^X$.

The number of hidden layers has to be chosen when designing the architecture of a neural network model. Neural networks with a single hidden layer, sufficiently large number of units $q_1 \in \mathbb{N}$ and a non-polynomial activation function have a so-called universal approximation property. However, adding more layers to the network may improve the model and allow the network to explore interactions of the covariates. Since we are interested in finding covariance structure beyond the ODP reserving model we decide to have three hidden layers, i.e. $H = 3$. The following hidden layers are defined as

$$\mathbf{z}^{(h),N} = \mathbf{z}^{(h)}(i,j) = \tanh(\mathbf{b}_h + \langle \mathbf{w}_h, \mathbf{z}^{(h-1)}(i,j) \rangle) \in \mathbb{R}^{q_h}, \tag{40}$$

where $\mathbf{b}_h \coloneqq \mathbf{b}_h^N$ and $\mathbf{w}_h \coloneqq \mathbf{w}_h^N$, and

$$\mathbf{z}^{(h),X} = \mathbf{z}^{(h)}(i,j,k) = \tanh(\mathbf{b}_h + \langle \mathbf{w}_h, \mathbf{z}^{(h-1)}(i,j,k) \rangle) \in \mathbb{R}^{q_h}, \tag{41}$$

where $\mathbf{b}_h = \mathbf{b}_h^X$, $\mathbf{w}_h = \mathbf{w}_h^X$, $\mathbf{b}_h \in \mathbb{R}^{q_h}$ are the vectors of biases and $\mathbf{w}_h \in \mathbb{R}^{q_{h-1} \times q_h}$ the weight matrices. The hidden layers have $q_1 = 20$, $q_2 = 15$ and $q_3 = 10$ units. We have chosen the hyperbolic tangent activation function $f = \tanh$ since it will allow us to compute the gradients efficiently due to $f' = 1 - f^2$. In addition the property $f \in (-1, 1)$ prevents the neurons from exploding. The output layer is finally defined as

$$(i,j) \to \mu^{\mathrm{NN},N}(i,j) = \exp\{\mathbf{b}_{H+1} + \langle \mathbf{w}_{H+1}, \mathbf{z}^{(H),N} \rangle\} \tag{42}$$

and

$$(i,j,k) \to \mu^{\mathrm{NN},X}(i,j,k) = \exp\{\mathbf{b}_{H+1} + \langle \mathbf{w}_{H+1}, \mathbf{z}^{(H),X} \rangle\} \tag{43}$$

where $\mathbf{c}_{H+1} \in \mathbb{R}$ is a scalar intercept and $\mathbf{w}_{H+1} \in \mathbb{R}^{q_H}$ are the weights. We set $q_{H+1} = 1$ and use the exponential activation function.

2.3.3.4 Cross-Classified Neural Network Regression Model

Next we will connect the cross-classified structure in Section 2.3.3.2 to the neural network structure in Section 2.3.3.3 in order obtain the cross-classified neural network regression model. This results in

$$\begin{aligned}(i,j) \to \mu^N(i,j) &= \exp\{\alpha^N(i) + \beta^N(j) + \mathbf{b}_{H+1} + \langle \mathbf{w}_{H+1}, \mathbf{z}^{(H)} \rangle\} \\ &= \mu^{CC}(i,j)\mu^{\mathrm{NN}}(i,j)\end{aligned} \tag{44}$$

for the claim counts part and analogously for the claim amounts part

$$\begin{aligned}(i,j,k) \to \mu^X(i,j,k) = \exp\{&\log(N_{i,j}) + \alpha^X(i) + \beta^X(j) + \gamma^X(k) \\ &+ \mathbf{b}_{H+1} + \langle \mathbf{w}_{H+1}, \mathbf{z}^{(H)} \rangle\} \\ = N_{i,j}&\mu^{CC}(i,j,k)\mu^{\mathrm{NN}}(i,j,k)\end{aligned} \tag{45}$$

Hence, we obtain the network parameter $\hat{\Theta} = \{(\hat{\alpha}_i^Y)_i, (\hat{\beta}_j^Y)_j, (\mathbf{b}_h)_h, (\mathbf{w}_h)_h\}$ and for the claim amounts $\hat{\Theta}$ also includes $(\hat{\gamma}_k^X)_k$. Adding the cross-classified structure to the neural network structure is called a skip connection, i.e. we directly add the embedding layer to the more classical neural network structure in (42) and (43), see the green line in Figure 1.

Initialisation of the network parameter $\hat{\Theta}$ is the second crucial moment in this blended model structure. We want the model to start exactly at the ODP reserving model and thus choose the initial value $\hat{\Theta}_0$ as follows:

$$\hat{\boldsymbol{\alpha}}_i = \hat{\alpha}_i^Y,\ \hat{\boldsymbol{\beta}}_j = \hat{\beta}_j^Y,\ \hat{\boldsymbol{\gamma}}_k = \hat{\gamma}_k^X,\ \mathbf{b}_{H+1} = \hat{c}_m^Y \text{ and } \mathbf{w}_{H+1} = 0, \tag{46}$$

where $\hat{b}^Y$ is the intercept from the ODP reserving model in (4) and (6). Setting the weights $\mathbf{w}_{H+1}$ to zero and the bias term $\mathbf{b}_{H+1}$ to the intercept from the ODP reserving models

$\hat{c}_m^Y$ allows us to start the gradient descent algorithm exactly at the ODP estimates (4) and (6). Finally, we can write the neural network reserving model similarly as in (25)

$$\begin{cases} \hat{\mathbb{E}}[X_{i,j,k}|N_{i,j}] & = N_{i,j} \exp\{f^{\mathrm{NN},X}(i,j,k;\hat{\Theta})\} \\ & = N_{i,j}\mu^X(i,j,k) \\ \hat{\mathbb{E}}[N_{i,j}] & = \exp\{f^{\mathrm{NN},N}(i,j;\hat{\Theta})\} \\ & = \mu^N(i,j) \end{cases} \tag{47}$$

2.3.4 Double Neural Network Model

The idea of the double feed-forward neural networks is that the joint modelling of the claim counts and claim amounts in (47) should improve the prediction of the claim amounts. The output of this neural network model is a two-dimensional vector as we model both claim counts and claim amounts simultaneously. The architecture of the model is illustrated in Figure 2. We can divide the model structure into four main parts: embedding layers (blue boxes), claim counts regression function (green boxes), payout attention function (violet boxes) and claim amounts regression function (red boxes). In this section we follow closely the neural network model description in [4].

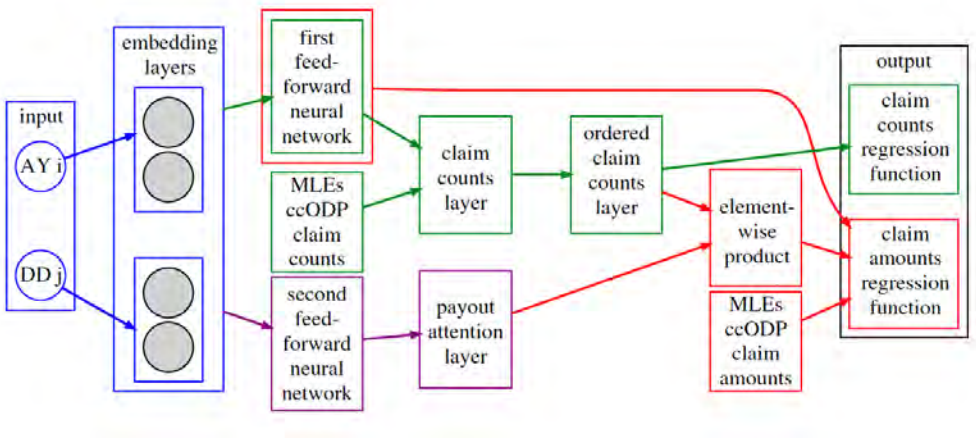

Figure 2: Illustration of the double neural network reserving model architecture.
Source: Figure 1 in [4].

2.3.4.1 Embedding Layers

The first part of the double neural network model is the same as in Section 2.3.3.1, i.e. we embed the ODP regression coefficients into input variables for the double neural network model:

$$\boldsymbol{\alpha}(\cdot) : (1, ..., I) \ \to \mathbb{R}^2, \quad i \to \hat{\boldsymbol{\alpha}}(i) = (\hat{\alpha}_i^N, \hat{\alpha}_i^X) \tag{48}$$

$$\boldsymbol{\beta}(\cdot) : (1, ..., J) \ \to \mathbb{R}^2, \quad j \to \hat{\boldsymbol{\beta}}(j) = (\hat{\beta}_j^N, \hat{\beta}_j^X) \tag{49}$$

and

$$\boldsymbol{\gamma}(\cdot) : (1, ..., K) \ \to \mathbb{R}, \quad k \to \hat{\boldsymbol{\gamma}}(k) = \hat{\gamma}_k^X \tag{50}$$

The embedding layers for the accident year, (48), and reporting delay, (49), are two-dimensional since both of these variables are included in the claim counts and payments models. As the payment delay is only relevant for the claim amounts part, the embedding layer for the payment delay, (50), is one-dimensional.

We choose to set the our embedding layers as non-trainable as we already have reasonable representation from the ODP reserving model. Another choice would be to train the embedding layers as well but since the starting values are already reasonable this should not make a huge difference on the output.

2.3.4.2 Claim Counts Regression Function

The second part of our neural network model is the claim counts regression function which is in green in Figure 2. We feed the five-dimensional vector consisting of the embedding layers

$$\mathbf{z}^{(0),N} = \mathbf{z}^{(0)}(i, j, k) = (\hat{\boldsymbol{\alpha}}(i), \hat{\boldsymbol{\beta}}(j), \hat{\boldsymbol{\gamma}}(k)) = (\hat{\alpha}_i^N, \hat{\alpha}_i^X, \hat{\beta}_j^N, \hat{\beta}_j^X, \hat{\gamma}_k^X) \tag{51}$$

into the first feed-forward neural network. Notice that unlike in (38), here we feed the also the embedded variables for the claim amounts part into the first hidden layer $\mathbf{z}^{(0),N}$, since we model the claim counts and payments jointly. The following layers are defined as

$$\mathbf{z}^{(h),N} = \mathbf{z}^{(h)}(i, j, k) = \tanh(\mathbf{b}_h + \langle \mathbf{w}_h, \mathbf{z}^{(h-1)}(i, j, k) \rangle) \in \mathbb{R}^{q_h} \tag{52}$$

where $\mathbf{b}_h$ denotes the bias and $\mathbf{w}_h$ denotes the weights of the hth hidden layer. Here as well, we have chosen the hyperbolic tangent activation function due to same argumentation as in Section 2.3.3.4. We define the output of the first feed-forward neural network for given $(i, j, k) \in \{1, ..., I\} \times \{1, ..., J\} \times \{1, ..., K\}$ as

$$\mathbf{z}^{(H),N} = \mathbf{z}^{(H)}(i, j, k) = \tanh(\mathbf{b}_H + \langle \mathbf{w}_H, \mathbf{z}^{(H-1)}(i, j, k) \rangle) \in \mathbb{R}^{q_H} \tag{53}$$

where $H = 2$ and q_H is the number of neurons in the output layer. We follow [4] and choose $q_1 = 30$ and $q_2 = 25$ neurons for the two hidden layers. Since neural networks are easily overfitted we use a dropout rate of 20 % at every layer h to prevent this. Moreover, we include L2-regularisation for each hidden layer with regularisation parameter $\lambda = 0.001$.

For the simple neural network in Section 2.3.3.4 we modelled $\mathbb{E}[N_{i,j}]$ individually for a given input (i, j) and obtained a one-dimensional output vector in (44). Here instead, we consider a representation of the whole vector $(\mathbb{E}[N_{i,0}], ..., \mathbb{E}[N_{i,J}])$ simultaneously. Thus, we

define $\mathbf{s}$ as the $(J+1)$-dimensional claim counts layer that follows after the output layer $\mathbf{z}^{(H),N}$ in the first feed-forward network. This layer is defined as

$$\mathbf{s} = \mathbf{s}(i,j,k) = (s_0(i,j,k), ..., s_J(i,j,k)), \tag{54}$$

where

$$s_l(i,j,k) = \exp\{\hat{\alpha}_i^N + \hat{\beta}_l^N + \mathbf{b}_l^{\mathbf{s}} + \langle \mathbf{w}_l^{\mathbf{s}}, \mathbf{z}^{(H)}(i,j,k)\rangle\} \tag{55}$$

for all $l = 0, ..., J$. The bias and the weights of the lth neuron of the claim counts layer $\mathbf{s}$ are denoted by $\mathbf{b}_l^{\mathbf{s}} \in \mathbb{R}$ and $\mathbf{w}_l^{\mathbf{s}} \in \mathbb{R}^{q_K}$. Notice that similarly as in (44), we have used a skip-connection to connect the ODP reserving model parameters $\hat{\alpha}_i^N$ and $\hat{\beta}_j^N$ directly to the neural network structure in the claim counts layer $\mathbf{s}$. By using initialisation

$$\mathbf{b}_l^{\mathbf{s}} = \hat{c}_m^N \quad \text{and} \quad \mathbf{w}_l^{\mathbf{s}} = \mathbf{0} \in \mathbb{R}^{q_H} \tag{56}$$

for all $l = 0, ..., J$, we have

$$\begin{aligned} \mathbf{s}(i,j,k) &= (\exp\{\hat{c}_m^N + \hat{\alpha}_i^N + \hat{\beta}_0^N\}, ..., \exp\{\hat{c}_m^N + \hat{\alpha}_i^N + \hat{\beta}_J^N\}) \\ &= (\hat{\mu}_{i,0}^N, ..., \hat{\mu}_{i,J}^N). \end{aligned} \tag{57}$$

Hence, the starting point for the claim counts layer is exactly the estimates $\hat{\mu}_{i,0}^N, ..., \hat{\mu}_{i,J}^N$ of $\mathbb{E}[N_{i,0}], ..., \mathbb{E}[N_{i,J}]$ from the ODP reserving model.

The expected claim amounts $\mathbb{E}[X_{i,j,k}]$ should depend on the number of observed claims $N_{i,j},$, i.e. $\mathbb{E}[X_{i,j,k}|N_{i,j}]$. Yet, the number of claims at times $i+j > I$ is unknown. Therefore, to avoid a break between training and prediction we use the predicted number of claims in the claim counts layer $\mathbf{s}$ for modelling the claim payments, i.e. we require the expected claim payment $\mathbb{E}[X_{i,j,k}]$ to depend on $s_0(i,j,k), ..., s_J(i,j,k)$ in (54).

Since we require $\mathbb{E}[X_{i,j,k}]$ to depend on the claim counts layer $\mathbf{s}$, it is important to ensure that only the claim counts $N_{i,j}$ have an effect on the expected claim payments $\mathbb{E}[X_{i,j,k}]$ and not the claim amounts after $i+j$, i.e. $N_{i,j+1}$. Moreover, we believe that the claim counts $N_{i,0,k}$ should have a similar effect on the claim amounts $X_{i,0,k}$ as the claim counts $N_{i,j,k}$ have on $X_{i,j,k}$ for all $j = 0, ..., J$. As the model is defined now, the claim counts $N_{i,0,k}$ would have a similar effect on $X_{i,0,k}$ and $X_{i,J,k}$ which may cause too high predicted reserves since the claim payments and claim counts are often larger on the upper triangle than on the lower.

This leads us to the next step of the double neural network model, namely the ordered claim counts layer $\mathbf{s}_{\text{ord}}$ that is defined as follows

$$\mathbf{s}_{\text{ord}} = \mathbf{s}_{\text{ord}}(i,j,k) = (s_j(i,j,k), s_{j-1}(i,j,k), ..., s_0(i,j,k), 0, ..., 0) \in \mathbb{R}^{J+1}. \tag{58}$$

Thus, we take the first $j+1$ neurons of the claim counts layer $\mathbf{s}$ and set them in the reverse order. In order to obtain a $(J+1)$-dimensional vector, we fill the rest of the elements in the vector with $J-j$ zeros.

Now that we have ordered the claim counts layer we can finally define the claim counts regression function as

$$(i,j) \mapsto \hat{\mu}^N(i,j,k) = s_j(i,j,k) = \exp\{\hat{\alpha}_i^N + \hat{\beta}_j^N + b_j^{\mathbf{s}} + \langle \mathbf{w}_j^{\mathbf{s}}, \mathbf{z}^{(H)}(i,j,k)\rangle\} \tag{59}$$

Note that we only need the first neuron $s_j(i,j,k)$ of the ordered claim counts layer $\mathbf{s}_{\text{ord}}$ to model the expected number of claims at time $i+j$. With initialisation as in (56), the claim counts regression function starts at exactly the ODP reserving model estimates

$$\hat{\mu}^N(i,j,k) = \exp\{\hat{c}_m^N + \hat{\alpha}_i^N + \hat{\beta}_j^N\} = \hat{\mu}_{i,j,k}^N \tag{60}$$

Remark that in (59) we do not have the intercept $\hat{c}_m^N$ from the ODP reserving model. Only after initialisation as in (56) we retrieve the $\hat{c}_m^N$ which allows us to start exactly at the ODP estimates.

2.3.4.3 Payout Attention Layer

The third part of the double neural network reserving model is the payout attention layer (see violet parts in Figure 2). The first step in this part is to define the second feed-forward network to capture the influence of the dependence of the accident year and reporting delay of past reported claims on the current claim amounts. The output from this is the payout attention layer that will be multiplied with the claim counts from the ordered claim counts layer $\mathbf{s}_{\text{ord}}$.

The second feed-forward network could be set up in multiple different ways but for simplicity we re-use the architecture from the first feed-forward network from Section 2.3.4.2. Hence, we have the same number of hidden layers, $H = 2$, the hyperbolic tangent activation function and the same number of neurons q_h. Moreover, we regulate the layers with 20 % dropout rate and L2-regularisation with $\lambda = 0.001$.

Similarly as in (51), we feed the five-dimensional vector consisting of the embedding layers in (48)-(50) into the first layer of the second feed-forward network:

$$\begin{aligned} \mathbf{z}^{(0),X} = \mathbf{z}^{(0)}(i,j,k) &= (\hat{\boldsymbol{\alpha}}(i), \hat{\boldsymbol{\beta}}(j), \hat{\boldsymbol{\gamma}}(k)) \\ &= (\hat{\alpha}_i^N, \hat{\alpha}_i^X, \hat{\beta}_j^N, \hat{\beta}_j^X, \hat{\gamma}_k^X). \end{aligned} \tag{61}$$

The following hidden layers in the network are defined as

$$\mathbf{z}^{(h),X} = \mathbf{z}^{(h)}(i,j,k) = \tanh(\mathbf{b}_h + \langle \mathbf{w}_h, \mathbf{z}^{(h-1)}(i,j,k)\rangle) \in \mathbb{R}^{q_h}. \tag{62}$$

The output from the second feed-forward network for $h = H$ is mapped to the $(J+1)$-dimensional vector that we call the payout attention layer. The name payout attention layer comes from attention mechanisms used e.g. in modelling natural languages, see [1] and [11]. The idea is that the layer focuses only on some relevant inputs and not all of them. We define the payout attention layer $\mathbf{a}$ as

$$\mathbf{a} = \mathbf{a}(i,j,k) = (a_0(i,j,k), ..., a_J(i,j,k)), \in \mathbb{R}^{J+1}, \tag{63}$$

where

$$a_l(i,j,k) = \mathbf{b}_l^{\mathbf{a}} + \langle \mathbf{w}_l^{\mathbf{a}}, \mathbf{z}^{(H),X}(i,j,k) \rangle \tag{64}$$

for all $l = 0, ..., J$ where $\mathbf{b}_l^{\mathbf{a}} \in \mathbb{R}$ denotes the bias of the lth neuron of the payout attention layer $\mathbf{a}$ and $\mathbf{w}_l^{\mathbf{a}}$ denotes the weights. Notice that instead of choosing an exponential activation function for the output layer as in our previous networks, see (44), (45) and (55), we choose a linear activation function. As before, the initialisation for the payout attention is important for the model structure which is set up as follows

$$\mathbf{b}_l^{\mathbf{a}} = 1 \quad \text{and} \quad \mathbf{w}_l^{\mathbf{a}} = \mathbf{0} \in \mathbb{R}^{q_H} \tag{65}$$

for all $l = 0, ..., J$. Thus the payout attention layer $\mathbf{a}$ is initially given by

$$\mathbf{a} = \mathbf{a}(i,j,k) = (1, ..., 1), \in \mathbb{R}^{J+1}. \tag{66}$$

The payout attention layer will be multiplied with the ordered claim counts layer $\mathbf{s}_{\text{ord}}$ defined in (58). Hence, with this initialisation we begin the gradient descent algorithm by assuming that the effect of past reported claim is the same across accident years and reporting / payment delays. Then we let the network explore and learn possible dependencies during several iterations.

2.3.4.4 Claim Amounts Regression Function

The final part of the double neural network model is the claim amounts regression function (see the parts in red in Figure 2). The first step is to calculate the element-wise (Hadamard) product between the ordered claim counts layer $\mathbf{s}_{\text{ord}}$ and the payout attention layer $\mathbf{a}$ and concatenate this with the output layer $\mathbf{z}^{H,N}$ from the first feed-forward neural network in (53). More formally

$$\mathbf{r} = \mathbf{r}(i,j,k) = (\mathbf{z}^H(i,j,k), \mathbf{a}(i,j,k) \odot \mathbf{s}_{\text{ord}}(i,j,k)) = (\mathbf{z}^{H,N}, \mathbf{a} \odot \mathbf{s}_{\text{ord}}), \tag{67}$$

where $\odot$ denotes the Hadamard product. Notice that we have applied batch normalisation, that is introduced in [9], to transform the values of the neurons in the ordered claim counts layer $\mathbf{s}_{ord}$ close to 0. The motivation for batch normalisation is that the distribution of the inputs of each layer varies during the training as the parameters from the previous layers changes for each iteration. Batch normalisation regulates the inputs and hence accelerates the gradient descent algorithm. The normalisation is beneficial for this complex network structure.

Finally, the concatenated layer $\mathbf{r}$ is mapped to the claim amounts regression function $\mu^X(\cdot,\cdot,\cdot)$ given by

$$(i,j,k) \mapsto \hat{\mu}^X(i,j,k) = \exp\{\hat{\alpha}_i^X + \hat{\beta}_j^X + \hat{\gamma}_k^X + \mathbf{b}_m^X + \langle \mathbf{w}_m^X, \mathbf{r}(i,j,k) \rangle\}, \tag{68}$$

where $\mathbf{b}_m^X \in \mathbb{R}$ and $\mathbf{w}_m^X \in \mathbb{R}^{q_H+J+1}$ denote the bias and the weights of the output neuron $\hat{\mu}^X(i,j,k)$ respectively. Note that the modelled claim amount $\mu^X(i,j,k)$ depends on the

regularised claim counts $s_j(i,j,k),...,s_0(i,j,k)$ up to time $i+j$ through the ordered claim counts layer $\mathbf{s}_{\text{ord}}$. Moreover, the reader might recognise the concept of skip-connection which we have used in (68) to connect that part of the embedding layer that refers to the claim amounts to the model, $\hat{\alpha}_i^X, \hat{\beta}_j^X, \hat{\gamma}_j^X$ directly to the neural network output.

In order to start the calibration exactly at the ODP reserving model estimates, we use the following initialisation

$$\mathbf{b}_m^X = \hat{c}_m^X \quad \text{and} \quad \mathbf{w}_m^X = \mathbf{0} \in \mathbb{R}^{q_H+J+1}. \tag{69}$$

Hence, the claim amounts regression in (68) starts at

$$\hat{\mu}^X(i,j,k) = \exp\{\hat{c}_m^X + \hat{\alpha}_i^X + \hat{\beta}_j^X + \hat{\gamma}_k^X\} = \hat{\mu}_{i,j,k}^X. \tag{70}$$

To summarise, the output from the double neural network model is the estimates for the claim counts from the claim counts regression function $\hat{\mu}^N$ in (59) and the estimates for the claim amounts from the claim amounts regression function $\hat{\mu}^X$ in (68).

2.3.5 Loss Functions

The ODP covariate estimates $\hat{\alpha}^Y$, $\hat{\beta}^Y$ and $\hat{\gamma}^X$ are found by minimising the Poisson deviance. We want to preserve the same structure in the neural network reserving models and hence, the loss function used in the simple neural network reserving model (see Section 2.3.3 and [5]) for the claim counts is the unscaled Poisson deviance given by

$$\begin{aligned} L^N((\mu^N(i,j))_{i,j}, \mathcal{D}_I^N) =& 2 \sum_{i+j\leq I} \mu^N(i,j) - N_{i,j} \\ &+ N_{i,j} \log\left(\frac{N_{i,j}}{\mu^N(i,j)}\right), \end{aligned} \tag{71}$$

where $\mathcal{D}_I^N$ denotes the upper triangle defined as $\mathcal{D}_I^N = \{N_{ij}, i = 1,...,i, j = 0,...,J, i+j \leq I\}$. For the claim amounts we have

$$\begin{aligned} L^X((\mu^X(i,j,k))_{i,j,k}, \mathcal{D}_I^X) =& 2 \sum_{i+j+k\leq I} \mu^X(i,j,k) - X_{i,j,k} \\ &+ X_{i,j,k} \log\left(\frac{X_{i,j,k}}{\mu^X(i,j,k)}\right), \end{aligned} \tag{72}$$

where $\mathcal{D}_I^X = \{X_{ijk}, i = 1,...,i, j = 0,...,J, k = 0,...,K, i+j+k \leq I\}$, see [5].

For the double neural network reserving model (see Section 2.3.4), we minimise the

scaled deviance for claim counts and claim amounts simultaneously as follows

$$\begin{aligned} L^{N,X}&((\mu^N(i,j,k),\mu^X(i,j,k))_{i,j,k},\mathcal{D}_I^N\cup\mathcal{D}_I^X;\hat{\phi}_m^N,\hat{\phi}_m^X) \\ &= 2\sum_{i+j+k\le I}\tilde{\mu}^N(i,j,k)-\tilde{N}_{i,j,k}+\tilde{N}_{i,j,k}\log\left(\frac{\tilde{N}_{i,j,k}}{\tilde{\mu}^N(i,j,k)}\right) \\ &\quad+\tilde{\mu}^X(i,j,k)-\tilde{X}_{i,j,k}+\tilde{X}_{i,j,k}\log\left(\frac{\tilde{X}_{i,j,k}}{\tilde{\mu}^X(i,j,k)}\right) \end{aligned} \tag{73}$$

where $\mathcal{D}_I^N\cup\mathcal{D}_I^X=\{N_{ij},X_{ijk},i=1,...,i,j=0,...,J,k=0,...,K,i+j+k\le I\}$ and

$$\tilde{\mu}^Y(i,j,k)=\mu^Y(i,j,k)/\hat{\phi}_m^Y \quad\text{and}\quad \tilde{Y}_{i,j,k}=Y_{i,j,k}/\hat{\phi}_m^Y \tag{74}$$

Hence, we minimise the sum of the claims counts loss ($Y=N$) and claim amounts loss ($Y=X$). Notice that we have scaled the predicted and observed values with the estimated over-dispersion parameter $\hat{\phi}^Y$. The loss function in (73) could be a weighted sum as well where we could set more weight to the claim amounts part. But since the claim counts are important for our model we choose equal weights for the claim counts and claims amounts parts, see [4].

The loss functions in (71), (72) and (73) for the simple and double neural network models are calculated iteratively using gradient descent method. We apply `rmsprop` which according to [7] has been empirically shown to be an effective method for training a deep neural network and it is widely used amongst practitioners. This method performs especially well in non-convex cases since it sets the gradient accumulation into an exponentially weighted moving average. RMSProp uses a hyperparameter ρ that controls the length of the weighted average and therefore discards the information from past extremes so that it can converge rapidly after finding a convex bowl. The generic RMSProp optimisation

algorithm is as follows

Algorithm 2: The RMSProp algorithm

Require: Global learning rate ϵ, decay rate ρ

Require: Initial parameter $\boldsymbol{\theta}$

Require: Small constant δ, usually 10^{-6}, used to stabilise division by small numbers

Initialise accumulation variables $\mathbf{r} = 0$

while *stopping criterion not met* **do**

Sample a minibatch of m examples from the training set $\{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(m)}\}$ with corresponding targets $\mathbf{y}^{(i)}$.

Compute gradient: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)})$.

Accumulate squared gradient: $\mathbf{r} \leftarrow \rho \mathbf{r} + (1 - \rho)\mathbf{g} \odot \mathbf{g}$.

Compute parameter update: $\Delta\boldsymbol{\theta} = -\frac{\epsilon}{\sqrt{\delta + \mathbf{r}}} \odot \mathbf{g}$. $\left(\frac{\epsilon}{\sqrt{\delta + \mathbf{r}}} \quad \text{applied element-wise}\right)$

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$.

end

For the implementation of the neural network reserving models we have used the `keras`-package in `R`. Hence, we refer to the documentation of the package for more details of the implementation of RMSProp algorithm in `keras`.

The double neural network model has total of

$$2\left(\sum_{h=1}^{H+1}(q_{h-1}+1)q_h\right) + 2(J+1) + (q_H + J + 1) + 1 \tag{75}$$

trainable parameters, where $q_0 = 5$ and $q_{H+1} = 12$. The simple neural network for claim counts part has

$$\left(\sum_{h=1}^{H+1}(q_{h-1}+1)q_h\right) + 1 \tag{76}$$

trainable parameters if we set the input parameter to non-trainable where $q_0 = 2$ and $q_{H+1} = 1$. Otherwise we will have to add $I + J$ to equation (76). For the claim payments part we have

$$\left(\sum_{h=1}^{H+1}(q_{h-1}+1)q_h\right) + 1 + 1 \tag{77}$$

trainable parameters with non-trainable inputs. Analogously, if we have trainable inputs then we have $I + J + K$ more trainable parameters. As we can see, the number of trainable parameters in the neural network models increases rather quickly as the number of hidden layers and units increases which make them easy to overfit. For example for the simulated data set in Section 3 with 12 accident years the number of trainable parameters in the double neural network model will be 2 596.

2.4 Conditional Mean Squared Error of Prediction

In this section we present the estimation of the conditional mean squared error of prediction. We follow closely [10].

The conditional mean squared error of prediction is given by

$$\begin{aligned} \mathrm{MSEP}(R, \hat{R}|N_0) &:= \mathbb{E}\left[(R-\hat{R})^2|N_0\right] \\ &= \mathrm{Var}(R|N_0) + (R - \mathbb{E}[\hat{R}|N_0])^2 \end{aligned} \tag{78}$$

Given the ODP assumptions, we can estimate the conditional mean squared error of prediction (MSEP) semi-analytically. The variance part of the MSEP can be calculated analytically and the estimation error part with bootstrap.

It follows from the ODP assumptions that

$$\mathrm{Var}(X_{i,j,k}|N_0) = \mathrm{Var}(X_{i,j,k}|N_{i,j}) = \phi^X \mu^X_{i,j,k} N_{i,j}. \tag{79}$$

where $i + j \leq I$. Since the claim amounts, the $X_{i,j,k}$'s, are independent given $N_{i,j}$ the RBNS variance for accident year i is given by

$$\begin{aligned} \mathrm{Var}(R_i^{\mathcal{R}}|N_0) &= \sum_{j=0}^{I-i} \sum_{k>I-i-j} \mathrm{Var}(X_{i,j,k}|N_0) \\ &= \phi^X \sum_{j=0}^{I-i} \sum_{k>I-i-j} \mu^X_{i,j,k} N_{i,j} \end{aligned} \tag{80}$$

The RBNS variances can be then summed over all accident years to obtain the total RBNS variance.

The process variance for the IBNR reserves is a bit more challenging since we are not conditioning on the claim amounts, the $X_{i,j,k}$'s, on the claim counts, the $N_{i,j}$'s. For

accident year i this is

$$\begin{aligned}
\mathrm{Var}(R_i^{\mathcal{I}}|N_0) &= \sum_{j=I-i+1}^{I-i} \mathrm{Var}\left(\sum_k X_{i,j,k}\right) \\
&= \sum_{j=I-i+1}^{I-i} \left(\mathbb{E}\left[\mathrm{Var}\left(\sum_k X_{i,j,k}|N_{i,j}\right)\right] + \mathrm{Var}\left(\mathbb{E}\left[\sum_k X_{i,j,k}|N_{i,j}\right]\right)\right) \\
&= \sum_{j=I-i+1}^{I-i} \left(\mathbb{E}\left[\mathrm{Var}\left(\sum_k X_{i,j,k}|N_{i,j}\right)\right] + \mathrm{Var}\left(\sum_k \mu_{i,j,k}^X N_{i,j}\right)\right) \\
&= \sum_{j=I-i+1}^{I-i} \left(\mathbb{E}\left[\sum_k \phi^X \mu_{i,j,k}^X N_{i,j}\right] + \left(\sum_k \mu_{i,j,k}^X\right)^2 \phi^N \mu_{i,j}^N\right) \\
&= \sum_{j=I-i+1}^{I-i} \left(\phi^X + \phi^N \sum_k \mu_{i,j,k}^X\right) \mu_{i,j}^N \sum_k \mu_{i,j,k}^X
\end{aligned} \tag{81}$$

As for the RBNS variances, the IBNR variances can be summed over the accident years to obtain the total process variance.

For the estimation error part of MSEP we simulate new in-data samples from a Poisson distribution using the following property

$$N_{i,j}/\phi^N \sim \mathrm{Po}(\mu_{i,j}^N/\phi^N) \tag{82}$$

then

$$N_{i,j} \sim \mathrm{ODP}(\mu_{i,j}^N, \phi^N). \tag{83}$$

Hence we can simulate data from a Poisson distribution with mean $\mu_{i,j}^N/\phi^N$ and then multiply the observations with the over-dispersion parameter ϕ to obtain a random sample from $\mathrm{ODP}(\mu_{i,j}^N, \phi^N)$.

The over-dispersion parameters, the ϕ^Y's, are based on the Pearson statistics defined in [12]. For the claim counts part this is

$$\hat{\phi}^N := \frac{1}{n - p_N} \sum_{i+j\leq I} \frac{(N_{i,j} - \hat{\mu}_{i,j}^N)^2}{\hat{\mu}_{i,j}^N} \tag{84}$$

where p_N is the number of the parameters ($p_N = 2I-1$) and n is the number of observations in the upper left triangle. For the claim amounts part of the estimated over-dispersion parameter $\hat{\phi}^X$ is given by

$$\hat{\phi}^X := \frac{1}{n - p_X} \sum_{i+j+k\leq I} \frac{(X_{i,j,k} - \hat{\mu}_{i,j,k}^X N_{i,j})^2}{\hat{\mu}_{i,j,k}^X N_{i,j}} \tag{85}$$

where n is the sample size of the $X_{i,j,k}$'s and p_X is the number of parameters ($p_X = 3I-2$).

Now that we have the variance part of the MSEP in (78) defined, we proceed to the estimation error part $(R-\mathbb{E}[\hat{R}|N_0])^2$. This is computed with the predicted RBNS and IBNR reserves $\hat{R}^{\mathcal{R}}, \hat{R}^{\mathcal{I}}$ and the bootstrapped reserves $\hat{R}^{\mathcal{R},*}_{(b)}, \hat{R}^{\mathcal{I},*}_{(b)}$. More formally, the estimation error is given by

$$\frac{1}{B}\sum_{b=1}^{B}(\hat{R}^{\mathcal{R}} + \hat{R}^{\mathcal{I}} - (\hat{R}^{\mathcal{R},*}_{(b)} + \hat{R}^{\mathcal{I},*}_{(b)}))^2, \tag{86}$$

where B is the number of bootstrap iterations. The bootstrap reserves $\hat{R}^{\mathcal{R},*}_{(b)}, \hat{R}^{\mathcal{I},*}_{(b)}$ are estimated with the following algorithms:

Algorithm 3: Bootstrap RBNS reserves

1. *Estimation of parameters:* Estimate the payment part parameters of the model using the original data to get the estimators $\hat{\mu}^X_{i,j,k}$ and $\hat{\mu}^N$.
2. *Bootstrapping the data:* With the same claim counts $N_{i,j}$, generate new bootstrap aggregated payments $\{X^*_{i,j,k} : i+j+k \leq I\}$ by simulating from $\text{Po}(\hat{\mu}^X_{i,j,k}N_{i,j}/\hat{\phi}^X)$ and multiplying by $\hat{\phi}^X$.
3. *Bootstrapping the parameters:* Compute the estimators $\hat{\mu}^{X,*}_{i,j,k}$ using $\{N_{i,j} : i+j \leq I\}$ and the bootstrap data $\{X^*_{i,j,k} : i+j+k \leq I\}$.
4. *Bootstrapping the RBNS predictions:* Using the original incurred claim counts $\{N_{i,j} : i+j \leq I\}$ and the bootstrap parameters $\hat{\mu}^{X,*}_{i,j,k}$ compute the RBNS reserve prediction $\hat{R}^{R,*}$, according to (9).
5. *Monte Carlo approximation:* Repeat steps 2. - 4. B times to get an approximate bootstrap distribution of the RBNS reserve from the bootstrapped $\{\hat{R}^{R,*}_{(b)}\}_{b=1}^B$.

Algorithm 4: Bootstrap IBNR reserves

1. *Estimation of parameters:* Estimate the parameters of the model using the original data to get the estimators $\hat{\mu}^N_{i,j}$, $\hat{\mu}^X_{i,j,k}$, $\hat{\phi}^N$ and $\hat{\phi}^X$.
2. *Bootstrapping the data:* Generate new bootstrap data $\{N^*_{i,j} : i+j+k \leq I\}$ and $\{X^*_{i,j,k} : i+j+k \leq I\}$ by simulating the $X^*_{i,j,k}$s exactly as in step 2. of the RBNS algorithm and the $N^*_{i,j}$s by simulating from $\text{Po}(\hat{\mu}^N_{i,j}/\hat{\phi}^N)$ and multiplying by $\hat{\phi}^N$.
3. *Bootstrapping the parameters:* Compute the estimators $\hat{\mu}^{X,*}_{i,j,k}$ using $\{N_{i,j} : i+j \leq I\}$ and the bootstrap data $\{X^*_{i,j,k} : i+j+k \leq I\}$ and compute the estimators $\hat{\mu}^{N,*}_{i,j}$ using $\{N^*_{i,j} : i+j \leq I\}$.
4. *Bootstrapping the IBNR predictions:* Using the bootstrap parameters $\hat{\mu}^{N,*}_{i,j}$ and $\hat{\mu}^{X,*}_{i,j,k}$, compute the IBNR reserve prediction $\hat{R}^{I,*}$, according to (10).
5. *Monte Carlo approximation:* Repeat steps 2. - 4. B times to get an approximate bootstrap distribution of the IBNR reserve from the bootstrapped $\{\hat{R}^{I,*}_{(b)}\}_{b=1}^B$.

3 Simulation study

In this section we present the results from the simulation study performed in the software `R` where we have applied the machine learning models discussed in Section 2 to a simulated claims data set.

3.1 Data Description

Data used in the simulation study is the same as in [5] and [10]. The data consists of individual non-life insurance claim histories from 6 different lines of business (LoB) and is generated with the claims simulation machine from [6]. We use the following information in our analysis; accident year $\in \{1994, ..., 2005\}$, reporting delay $\in \{0, ..., 11\}$ and payment delay $\in \{0, ..., 11\}$ which is defined as the number of years after reporting the claim the payment incurs. There are approximately 250 000 claims for each LoB except for LoB 3 and 6 that have close to 100 000 claims respectively. As we can see from Table 1 most of the claims are RBNS claims and the outstanding reserves varies from around 17 million to 73 million.

LoB	N	Payment	RBNS %	Reserve
1	250 040	285 989	99.40	39 689
2	250 197	278 621	99.41	37 037
3	99 969	108 345	99.26	16 878
4	249 683	429 344	99.20	71 630
5	249 298	437 728	99.19	72 548
6	99 701	171 482	99.10	31 117

Table 1: Number of claims, total payments, percentage of the claims that are RBNS and outstanding reserve for each LoB (in thousands).

In Figure 3 (left) the average total payment per development year for each LoB is illustrated. Most of the payments are made during the first development year and the payment sizes declines rather fast during the first development years. The average payments are almost identical for the first two LoBs and LoBs 4 and 5.

The average claim cost per accident year is illustrated in Figure 3 (right). We can see that the first three LoBs have lower average claim cost compared to the last three LoBs but all LoBs have a positive trend in the average claim cost over the years. The cumulative claim payments for each LoB are found in Tables 12 - 17 in Appendix A.

For training the models we have divided the individual claims data into training and validation data that are approximately the same size (50/50) since the model parameters from the ODP model are sensitive to the volume of the data and these are the input weight to our neural network models. Data is thereafter aggregated by LoB, accident year, reporting delay and payment delay the same way as in [10]. Some of the aggregated

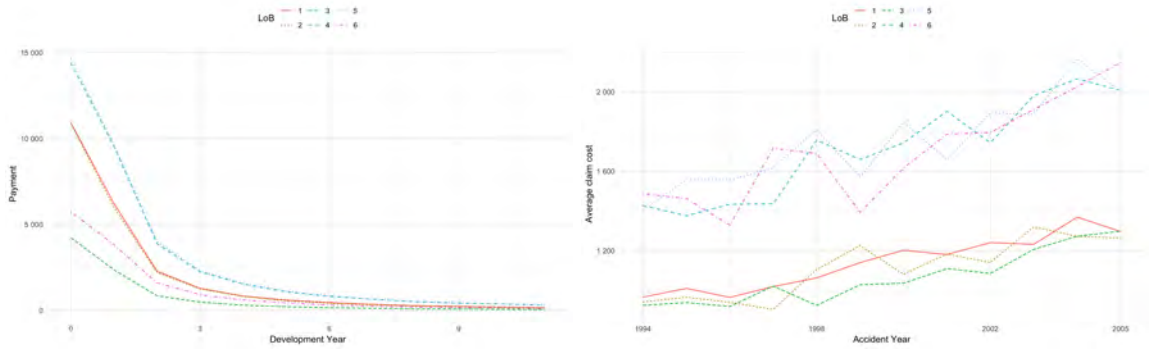

Figure 3: *Left:* Average payment per development year (in thousands). *Right:* Average claim cost over accident years.

payments are negative in the accident year, reporting and payment delay groups and since our models use the Poisson loss function we replace the negative payments with zeros. The training is done on the upper left triangles, $\mathcal{D}_I^Y$, and after the optimal hyper-parameters are found we predict the outstanding reserves and calculate the prediction error (out-of the sample error) on the lower triangles.

3.2 Results

We start by finding the optimal tuning parameters for our models and begin with GBM. Notice that we treat all the variables as numerical in the GBM models and therefore it is easy to add linear combinations of the variables. Hence we add an inflation factor to the model, defined as the sum of accident year and reporting delay. Standard values for the tuning parameters discussed in Section 2.2 in the `gbm`-function are: shrinkage 0.1, depth 1 i.e. no interactions, bagging factor 0.5 and at least 10 observations per leaf. We will try slightly different values for the tuning parameter to find the optimal ones for the claim counts and claim amounts models respectively. The training and validation loss plots are very similar for all of the LoBs. Thus we show the plots only for LoB 1 here and the remaining plots for the other LoBs are found in Appendix B.

Figure 4, top left corner, shows the training and validation loss for the claim counts model with shrinkage factor 0.1 (green and violet lines) and 0.01 (red and blue lines). The validation error would eventually be a bit lower with shrinkage factor 0.01 but it would require considerably more trees compared to setting the shrinkage factor to 0.1. We do not believe that the gain in the prediction ability is significant when using the smaller shrinkage factor 0.01 and therefore choose the simpler model with shrinkage 0.1.

The training and validation loss with at least 1 observation per node (green and violet lines) and 10 observations per node (red and blue lines) is illustrated in the top right corner of Figure 4. The validation loss curves are very similar despite the choice of the tuning

parameter. Since we model the claim count and amounts on aggregated data we have quite few observations in the upper left triangle. Especially for the claim counts we only have 78 observations per LoB. Thus we do not want to restrict the model to obtain at least 10 observations per node. Notice though that this can lead to that the average of the outcome is the outcome itself.

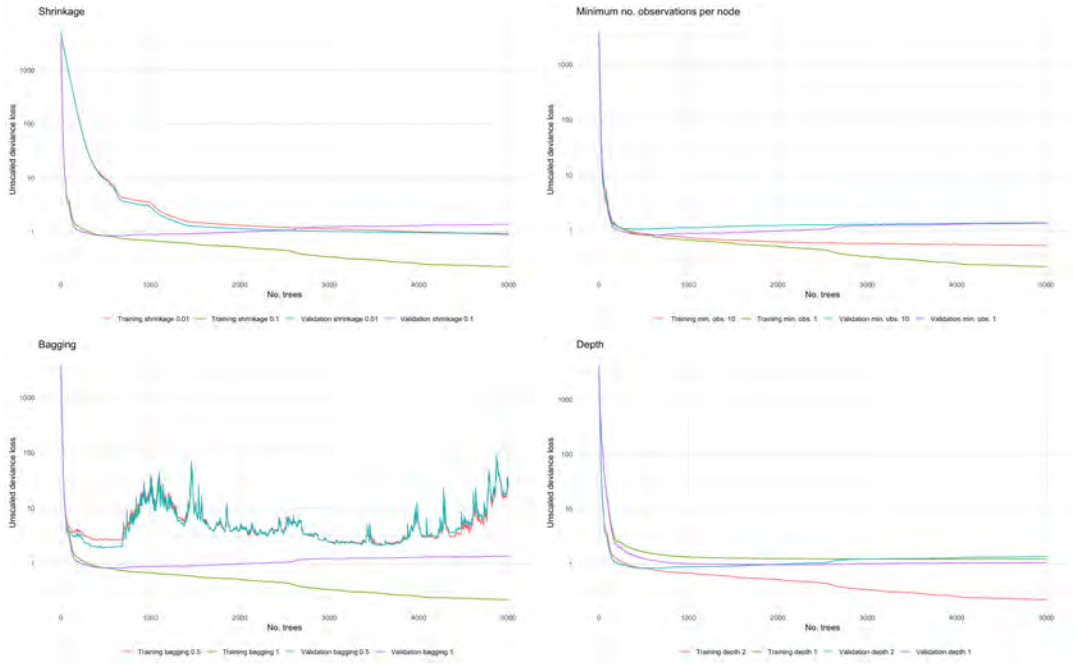

Figure 4: Training and validation error for the claim counts part of GBM for LoB 1. *Up left:* Shrinkage factor 0.1 and 0.01. *Up right:* Minimum observations per node 1 and 10. *Bottom left:* Bagging factor 0.5 and 1. *Bottom right:* Interaction depth 1 and 2.

Bagging leads to more volatile training and validation losses as is seen in the bottom left corner in Figure 4. We obtain a smaller and more stable loss when choosing bagging factor 1 i.e. we do not include bagging in the model. Finally, the training and validation loss for the claim counts model with tree interaction depth 2 (blue and red lines) and 1, i.e. no interactions, (violet and green lines) are illustrated in the bottom right of Figure 4. With interaction depth 2 the validation loss declines faster than with no interactions. Hence, we include interactions in the model.

Now that we have the tuned parameters for the claim counts part of the GBM we proceed with the claim amounts part. In the same way we compare the validations losses for two different choices of tuning parameters. The training and validation losses for the claim amounts GBM model LoB 1 are illustrated in Figure 5. In the top left corner, the red and blue lines shows the training and validation losses with shrinkage factor 0.1 and 0.01.

As we saw in Figure 4, with shrinkage factor 0.1 the validation loss reaches its minimum faster than with 0.01 as expected. Hence, we choose the larger value for shrinkage. The larger shrinkage factor leads also to 10 times faster computation time.

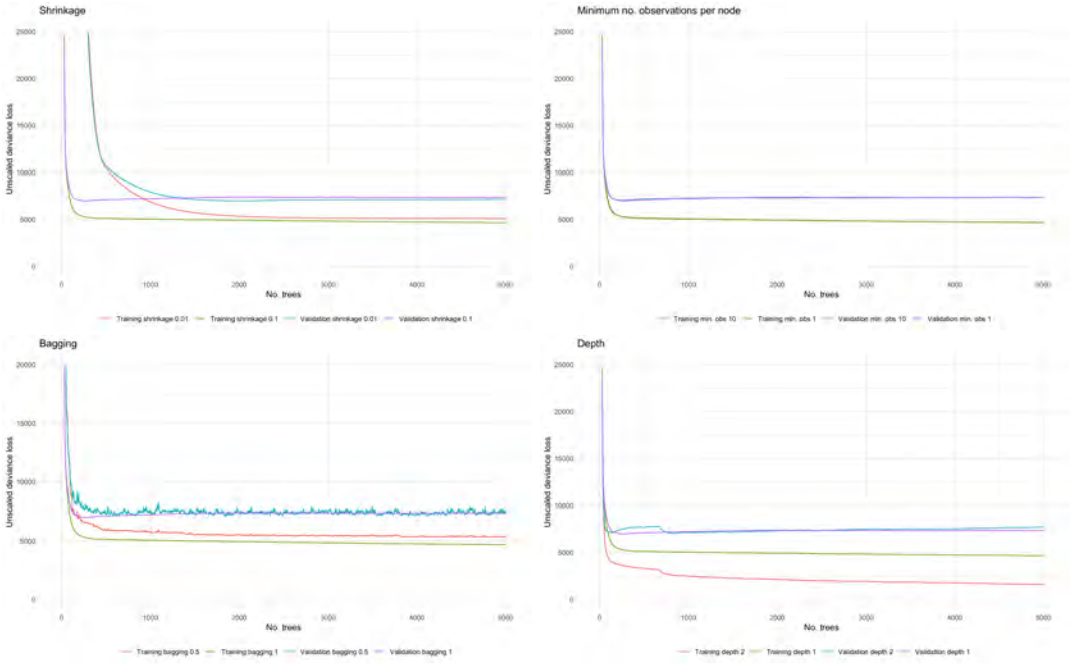

Figure 5: Training and validation error for the claim amounts part of GBM for LoB 1. *Up left:* Shrinkage factor 0.1 and 0.01. *Up right:* Minimum observations per node 1 and 10. *Bottom left:* Bagging factor 0.5 and 1. *Bottom right:* Interaction depth 1 and 2.

The training and validation loss for the model with at least 1 and 10 observation per node is shown in the top right corner of Figure 5. There is hardly any difference between the choices of the tuning parameter. Thus, with the same argumentation as for the claim counts part, we choose a minimum 1 observation per node due to the restricted amount of data.

In the bottom left corner, the red and blue lines shows the training and validation losses with bagging factor 0.5 and the green and violet lines for bagging factor 1, i.e. no bagging. The validation loss is very similar for both values of the bagging factor. As expected bagging of 0.5 yields a more volatile validation loss curve than without bagging. Moreover, the validation loss without bagging requires fewer trees at its minimum. Hence, we choose to not to include bagging in the model.

The training and validation losses for depth 1 (green and violet lines) and 2 (red and blue lines) are shown in the bottom right corner of Figure 5. The validation loss with interaction depth 2 first declines and then rises whereafter it decreases again to the same

level as with interaction depth 1, i.e. no interactions. We tested the fitting with both choices of interaction depth and noticed that the out-of the sample error is terrible when we include interactions, thus we decide to leave it out.

To summarise, the tuning parameters that we have chosen for the GBM models are as follows: bagging factor 1, i.e. no bagging, shrinkage 0.1, minimum observations per node 1, interaction depth 2 for the claim counts part and 1 for the claim amounts part. The final tuning parameter to be chosen for GBM is the number of trees.

In Figure 6 the training and validation losses for the standard tuning (red and blue lines) and for our tuning (green and violet lines) are shown for the claim counts part for all LoBs. It is clear that our tuning improves the fitting significantly compared to the standard tuning and hence we choose our tuning.

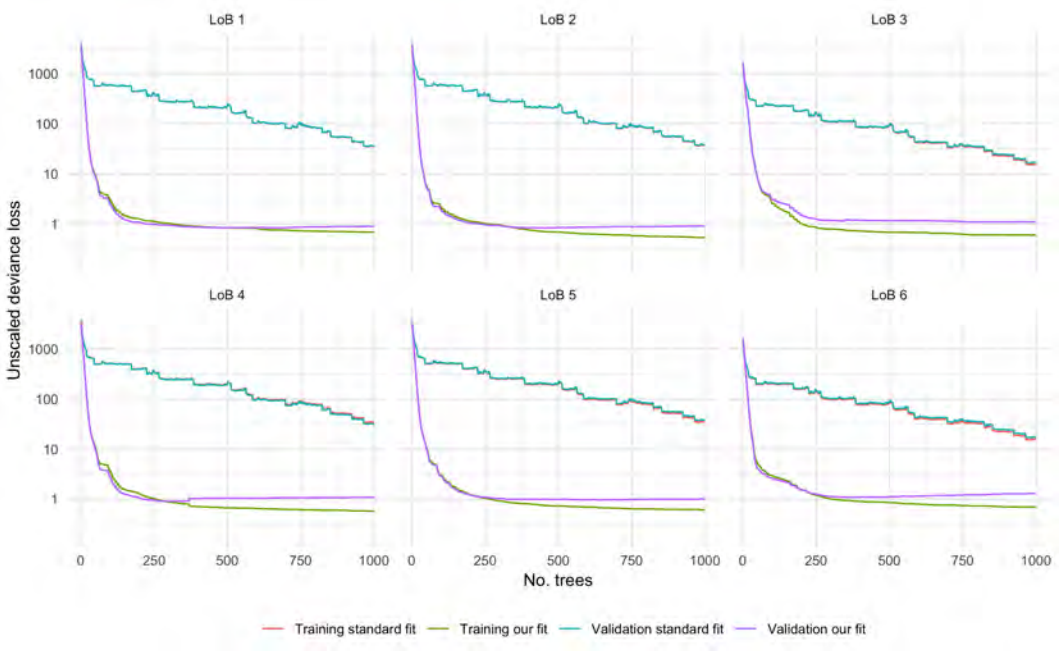

Figure 6: Training and validation error for GBM models for our fitting (green and violet line) and standard fitting (red and blue line). *Up:* Claim counts part. *Down:* Claim amounts part.

The standard tuning gives quite similar results compared to our tuning for claim amounts part as is seen from Figure 7. As discussed before the bagging factor yields a bit more unstable losses compared to no bagging. In order to avoid overfitting the models we use early stopping by choosing the number of trees where the validation loss is at its minumum. As we can see the models converge rather fast and do not require a large number of trees. All of the LoBs need less than 400 trees for the claim amounts part and less

than 800 for the claim counts part. The exact number of trees for each LoB and model is summarised in Table 2.

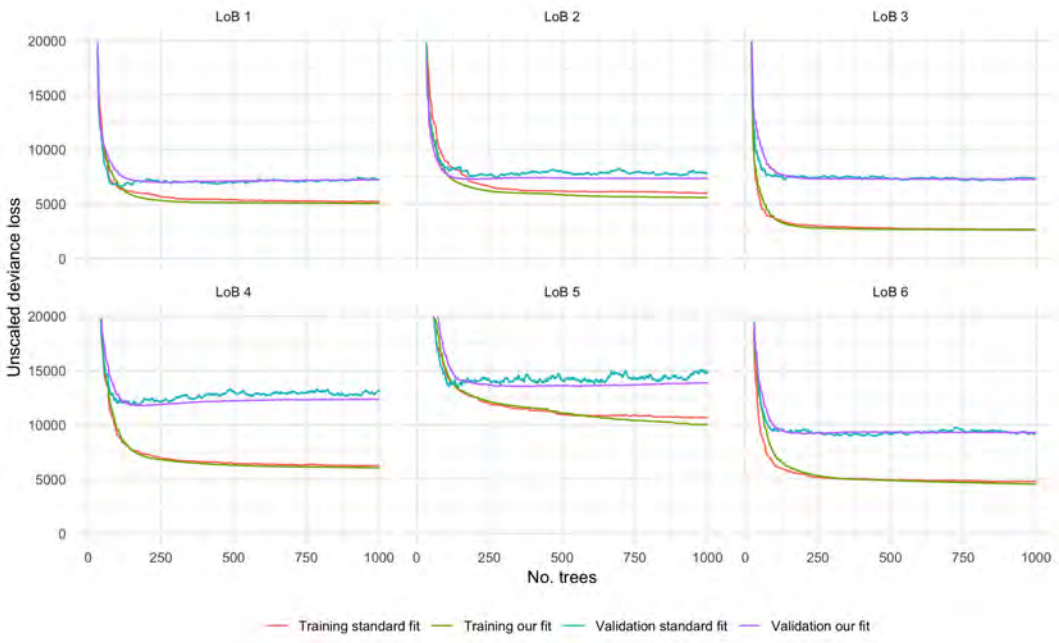

Figure 7: Training and validation error for GBM models for our fitting (green and violet line) and standard fitting (red and blue line). *Up:* Claim counts part. *Down:* Claim amounts part.

We now proceed with the fitting of the neural network reserving models presented in Section 2.3. The only tuning parameter we consider here is the number of epochs. Models are fitted for 10 000 epochs and the training and validation losses are illustrated in Figure 8. The number of epochs is then chosen by a simple central moving average with window size 100. From the top of Figure 8 we can see that the claim counts part of the simple neural network model converges slowly and we might benefit from letting the model run more than 10 000 epochs, especially LoBs 1, 4 and 5. For LoBs 2 and there is hardly any decrease in the validation loss indicating that the predictions from the ODP model are sufficient. The training and validation losses for the payment part of the simple neural network model are shown in the middle of Figure 8. We can see that the model starts to overfit for LoBs 2, 3, 5 and 6 after less than 1 000 epochs as the validation loss starts to increase and the training loss decreases. LoBs 1 and 4 do not have similar increase in the validation loss and the minimum of the moving average with window size lands to more than 9 000 epochs.

At the bottom of Figure 8 we can see the ODP model does not improve significantly in

the double neural network model for LoB 5 as the validation loss is a straight line through all 10 000 epochs. For LoB 6 we can notice that the validation error first drops and then starts to increase again which indicates that we should choose the number of epochs were the validation loss is at its lowest, otherwise we could have a model that overfits data. For most of the LoBs (except for 3 and 6) we choose number of epochs around 7 000. Table 2 summarises the number of trees and epochs needed for the models.

		LoB					
Model	Type	1	2	3	4	5	6
GBM	Count	635	455	778	304	632	373
GBM	Amount	279	200	240	171	384	204
Simple NN	Count	9 950	9 865	9 266	9 893	4 566	2 567
Simple NN	Amount	9 560	274	421	9 899	392	326
Double NN	Claims	7 549	6 285	2 115	7 423	7 935	944

Table 2: Number of trees and epochs used in the machine learning models.

Now that we have optimised all the tuning parameters for the models, we proceed with examining how well the models actually predict the outstanding reserves, i.e. data that the models have not seen before. Hence, the models are fitted again on the entire upper triangle (training and validation data) and predictions are made on the lower triangle.

Table 3 shows the actual outstanding reserves for each LoB and the predicted reserves together with the relative biases for Chain-Ladder, ODP reserving model, GBM and the two neural network models. We can notice that GBM model performs well as the highest bias is only 3.20 %. Especially excellent performance is shown for LoB 1 where the bias is only 0.02 % which can be compared to -2.82 % for Chain-Ladder. The neural network models have more variation in the performance, the simple neural network has bias 0.07 % at its lowest and -9.02 % at its highest. This model gives the most accurately predicted reserves for LoB 4. For LoBs 3 and 5 we obtain a slightly worse prediction than for the ODP model and Chain-Ladder. Overall the double neural network model seems to perform better than the simple neural network.

We also fitted the simple neural network model with trainable input weights (TW) to test if the predictions would improve. As a result, the predictions indeed improved for most of the LoBs and especially for LoB 3 where the bias was already worst. The double neural network was also fitted with trainable input weights but this did not make any significant difference on the predictions. It should be noted that by letting the input weights to be trainable, the computation time of the network increases.

During the fitting procedure we found that the machine learning models can be sensitive to the division of training and validation sets. Hence, we fitted the models also by flipping the training and validation sets, i.e. we trained the model with our original validation data and then validated on the training data. The results are shown at the bottom of Table

3 (VT). For some models and LoBs the results are better than the original results, for example simple neural network model with LoBs 1-4 and 6. For other models the effect is the opposite, see GBM for LoB 5 where the relative bias in the original model is -1.54 % and when flipping the training data it increases to 12.04 %. Interestingly the double neural network model is the best predictor for most of the LoBs and outperforms GBM when flipping the training and validation data. Notice that the bias for several models and LoBs is the opposite for VT models which gives rise to selecting the average outstanding reserve as the prediction. The average predicted reserves are shown in the parenthesis in Table 3.

		LoB					
Model	Type	1	2	3	4	5	6
True	Reserve	39 689	37 037	16 878	71 630	72 548	31 117
CL	Reserve	38 569	35 460	15 692	67 574	70 166	29 409
CL	Bias %	-2.82	-4.26	-7.02	-5.66	-3.28	-5.49
ODP	Reserve	38 308	35 151	15 452	67 055	69 470	29 115
ODP	Bias %	-3.48	-5.10	-8.45	-6.39	-4.24	-6.44
GBM	Reserve	39 697	37 229	16 367	72 667	71 433	32 114
GBM	Bias %	**0.02**	**0.52**	**-3.03**	1.44	-1.54	**3.20**
Simple NN	Reserve	41 268	34 779	15 356	71 682	70 649	29 336
Simple NN	Bias %	3.98	-6.10	-9.02	**0.07**	-2.62	-5.73
Simple NN TW	Reserve	41 005	39 135	15 949	75 646	75 438	29 640
Simple NN TW	Bias %	3.31	5.66	-5.50	5.54	3.98	-4.75
Double NN	Reserve	40 029	35 959	15 686	69 509	72 512	30 047
Double NN	Bias %	0.85	-2.91	-7.06	-2.96	**-0.05**	-3.44
GBM VT	Reserve	39 925	35 624	17 240	68 333	81 279	30 348
GBM VT	Bias %	**0.59**	-3.82	**2.15**	-4.61	12.04	-2.47
		(0.31)	(-1.65)	(-0.44)	(-1.58)	(5.25)	(0.37)
Simple NN VT	Reserve	38 738	38 657	15 719	70 146	70 017	29 465
Simple NN VT	Bias %	-2.40	4.37	-6.86	-2.07	-3.49	-5.31
		(0.78)	(-0.86)	(-7.94)	(-1.00)	(-3.05)	(-5.52)
Simple NN TW VT	Reserve	41 704	38 882	17 583	73 925	74 483	31 701
Simple NN TW VT	Bias %	5.08	4.98	4.18	3.20	2.67	1.87
		(4.20)	(5.32)	(-0.66)	(4.40)	(3.32)	(-1.43)
Double NN VT	Reserve	38 577	35 822	15 580	70 522	71 818	30 615
Double NN VT	Bias %	-2.81	**-3.28**	-7.69	**-1.55**	**-1.01**	**-1.61**
		(-0.97)	(-3.09)	(-7.38)	(-2.25)	(-0.53)	(-2.53)

Table 3: True reserves and predicted reserves with relative biases for each LoB. The predicted reserves at the top of the table are from the models with the original split into training and validation data. At the bottom of the table are shown the predicted reserves when training is performed on the original validation data and validation on the training data (VT). The relative bias for the average predicted reserve is shown in the parenthesis. Simple neural network model is also fitted with trainable input weights (TW).

As we condition the expected claim amounts on the number of claims, the predicted number of claims are also of interest. The results from the claim counts models are shown in Table 4. GBM is the best predictor for almost all of the LoBs except for LoB 2 where the neural network models appears to have better predictions. Notice that the neural network models seem to improve the ODP predictions somewhat. The number of predicted claims are divided into RBNS claims and IBNR claims for the double neural network model since the RBNS reserve uses the predicted number of claims and not the actual. The biases for the number of RBNS claims are quite small though which seems reasonable since the model has seen the actual number of RBNS claims during the fitting.

		LoB					
Model	Type	1	2	3	4	5	6
True	Claims	1 490	1 479	740	1 992	2 008	893
ODP	Claims	1 712	1 719	835	2 398	2 412	1 092
ODP	Bias %	14.90	16.23	12.84	20.38	20.12	22.28
GBM	Claims	1 654	1 691	767	2 117	2 077	930
GBM	Bias %	**11.01**	14.33	**-3.65**	**6.28**	**3.44**	**4.14**
Simple NN	Claims	1 678	1 630	786	2 302	2 360	1 053
Simple NN	Bias %	12.62	**10.21**	6.22	15.56	17.53	17.92
Double NN	Claims IBNR	1 165	1 667	872	2 348	2 428	1 046
Double NN	Bias %	11.74	12.71	17.84	17.87	20.92	17.13
Double NN	Claims RBNS	228 729	228 420	93 126	227 763	228 278	91 299
Double NN	Bias %	-0.06	0.06	0.15	-0.06	0.09	-0.26

Table 4: True and predicted IBNR claim counts with relative biases for each LoB for the lower right triangle. The predicted RBNS claim counts are only shown for double neural network model. Since the other models use the actual number of claims for modelling of the RBNS reserves, the predicted IBNR claims are only of interest.

The advantage of dividing the development dynamics into reporting and payment delay is that it allows us to predict the RBNS and IBNR reserves separately. Table 5 shows the RBNS and IBNR reserves for each LoB. The relative biases for the IBNR reserves are worse than for the RBNS reserves as expected since the IBNR reserves are much smaller and the conditioned claim counts are the predicted ones. Here as well, we notice the good performance of the GBM model that predicts the IBNR reserves best for all of the LoBs, expect for LoB 3. The relative bias for LoB 5 with GBM is 2.15 % while the other models have relative bias between 25 % to 30 %.

The relative biases for the claim amounts per accident year and development year are shown in Figure 9 for Chain-Ladder, GBM, double and simple neural network reserving models. The results for the neural network models are similar to Chain-Ladder and it is clear that the models underestimate the reserves systematically on the lower triangle. GBM instead eliminates the systematic underestimation. Notice that the highest biases are observed for the last development years but the payments are very small meaning that

		LoB					
Model	Type	1	2	3	4	5	6
True	IBNR	1 597	1 538	603	3 594	2 739	1 048
ODP	IBNR	1 861	1 835	1 020	3 664	3 549	1 651
ODP	Bias %	16.53	19.31	69.15	**1.95**	29.57	57.54
GBM	IBNR	1 644	1 629	837	3 186	2 797	1 405
GBM	Bias %	**2.90**	**5.91**	**38.77**	-11.34	**2.15**	**34.07**
Simple NN	IBNR	1 803	1 712	929	3 433	3 519	1 578
Simple NN	Bias %	12.91	11.30	53.89	-4.48	28.52	50.61
Double NN	IBNR	1 890	1 813	1 004	3 691	3 437	1 736
Double NN	Bias %	18.34	17.84	66.32	2.68	25.49	65.68
True	RBNS	38 093	35 500	16 275	68 038	69 810	30 070
ODP	RBNS	36 447	33 317	14 432	63 392	65 922	27 464
ODP	Bias %	-4.32	-6.15	-11.32	-6.83	-5.57	-8.67
GBM	RBNS	38 053	35 600	15 529	69 481	68 636	30 709
GBM	Bias %	**-0.01**	**0.28**	**-4.58**	2.12	-1.68	**2.12**
Simple NN	RBNS	39 465	33 067	14 427	68 249	67 130	27 758
Simple NN	Bias %	3.60	-6.85	-11.35	**0.31**	-3.84	-7.69
Double NN	RBNS	38 139	34 146	14 682	65 818	69 075	28 311
Double NN	Bias %	0.12	-3.81	-9.78	-3.26	**-1.05**	-5.85

Table 5: True RBNS and IBNR reserves and predicted reserves with relative biases for each LoB.

the absolute biases are not large. The relative biases for the claim counts part are shown in Figure 10. Simple neural network predicts the number of claims almost exactly on the upper triangle. Chain-Ladder is the worst predictor for claim counts for development years 10 and 11 where the number of claims is very small. The machine learning models succeed to predict the number of claims better than Chain-Ladder.

In order to examine how well the models perform we have plotted the average total payment per development year and average claim cost per accident years for LoB 6 in Figure 11. The predicted average payments per development year are very close to the observed average payments and there is no clear difference between the models. Furthermore, the predicted average claim costs are very close to the true values for all the models except GBM where the average claim cost is more stable over the accident years. The reason why we choose to illustrate these for LoB 6 is that the average claim cost for GBM differs in comparison to the other models. For all the other LoB's all the models follows the true average claim cost closely.

During the fitting procedure we noticed that the neural network models are quite sensitive for the choice of seed as well. The results in Table 3 are generated with seed 75 and we were also interested in how much the predictions varies when changing the seed but still preserving the same number of epochs as in the original model. Thus, we fitted the double

neural network model for 20 different seeds to get a more stable prediction by taking the mean of the predicted reserves. The relative biases are illustrated in a box plot in Figure 12. There is indeed some fluctuation in the predictions, particularly for LoB 2 with seeds 65-84. LoB 1 have more variation in the predictions for seeds 1-20 than for seeds 65-84. Based on Figure 12 it might be beneficial to fit several models with different seeds and then choose the average prediction as the predictor of the outstanding reserve.

We were also interested in exploring how well the reserving models would perform if we reduce the training data. Thus, we fitted the models while removing the last diagonal from the training set. Notice that this method is only possible for the GBM reserving model since the neural network models require input values for each accident year, reporting delay and payment delay. As we remove the last diagonal we do not have input for the last accident year and hence neural network reserving models are not applicable. The GBM model still performs well even without the last diagonal in the training set with the following relative biases: 1.79 %, 0.90 %, -1.56 %, -3.21 %, -3.71 % and -1.97 %. In fact, the predictions are even better for LoB 3 and 6 without the last diagonal.

Figure 13 shows the relative bias when gradually removing some of the individual observations from the data. Before aggregating the individual claims per accident year, reporting delay and payment delay, the individual claims data is sorted per LoB and accident year. We removed systematically every other observation when reducing the number of individual observations to half, every third when removing one third and so on. GBM has more difficulties in predicting the outstanding reserves compared to the other models. Here we see that GBM fails especially for LoB 3 since the number of claims is quite small and therefore the division into groups by the covariates becomes hard. The neural network models are always smooth and hence they are able to fit the data better when the number of observations declines. Interestingly for LoB 5 there is a peak in the bias when 75 % of the observations are removed and yet when 90 % of the observations are removed the bias is close to 0 %.

So far we have used the relative bias when comparing and evaluating the performance of the reserving models. However, normally we do not know the true reserves which prevents us from computing the relative bias. In addition, we have only seen how well the models perform on one data set. As a consequence, we need to expand our tools for evaluating the models and therefore it might be interesting to look at the conditional MSEP that is computed according to (78). When calculating the conditional MSEP we assume that the predicted reserves are the true ones and bootstrap several new data sets that have similar characteristics as the original data set. MSEP balances the trade-off between variance and bias and a small value is desired.

Fitting the models to each new simulated data set requires considerably computation power which is why we have chosen to restrict the number of simulations. We have performed 1 000 simulations for all the models, except for the simple neural network model for which the fitting is remarkably slower than the other models. Thus, we have performed 100 simulations for the neural network reserving model. Table 6 summarises the MSEPs for

the reserving models together with the root of the process variances and estimation errors. The highest MSEPs are observed for GBM and the lowest for ODP and Chain-Ladder. The drawback of Chain-Ladder is though that it does not allow for separate modelling of the RBNS and IBNR reserves. As the GBM and neural network models are highly parametrised we can expect greater estimation error than for the ODP and Chain-Ladder reserving models. Notice that neural network has the smallest process variance.

Histograms of the predicted reserves for the simulated datasets for LoB 2 are illustrated in Figure 14 where the red lines show the predicted reserves for each reserving model and the black lines show the true reserves. The distributions of predicted reserves with Chain-Ladder and ODP reserving models are rather similar. Moreover, both of these models, as well as neural network reserving model, underestimates the true reserves whereas GBM overestimates. Recall that in our estimation of MSEP we assume that the predicted reserves are true ones. Chain-Ladder and ODP reserving model predictions are around the median of the distribution of the simulated predictions. GBM instead overestimates and neural network underestimates the predicted reserves severely.

Table 6 shows also the over-dispersion parameters for the claim counts ϕ^N and claim amounts ϕ^X. For the claim counts part the over-dispersion parameters are rather small and even smaller than 1 for GBM.

For the estimation of the MSEPs we have used parametric bootstrap simulations which does not catch model error since we assume that the predicted values are the true values. The MSEPs in Table 6 could also be compared to the unconditional MSEPs where instead of parametric bootstrap new data is simulated from the simulation machine in [6] with different seeds. However we expect the unconditional MSEPs to be close to the conditional MSEPs. By comparing these we could get an indication of the models true MSEPs.

	LoB					
MSEP	1	2	3	4	5	6
CL	1 120	1 287	480	2 195	2 000	953
ODP	1 017	1 158	617	1 706	2 353	1 162
GBM	1 746	2 699	1 263	5 877	4 597	3 052
NN	1 881	1 794	750	3 216	3 709	1 305
Process Variance	1	2	3	4	5	6
CL	587	715	245	1 110	1 000	491
ODP	560	633	330	877	1 208	600
GBM	683	751	410	1 030	2 031	746
NN	442	598	327	677	1 130	598
Estimation Error	1	2	3	4	5	6
CL	954	1 069	412	1 894	1 732	817
ODP	849	969	521	1 463	2 019	995
GBM	1 607	2 593	1 195	5 786	4 124	2 959
NN	1 828	1 691	675	3 144	3 533	1 160
Over-dispersion ϕ^X	1	2	3	4	5	6
CL	8.9	14.4	3.8	18.2	14.2	8.2
ODP	8.1	11.3	6.9	11.2	20.8	12.2
GBM	11.7	15.1	10.2	14.5	57.7	17.2
NN	4.6	10.2	6.9	6.2	17.9	12.0
Over-dispersion ϕ^N	1	2	3	4	5	6
ODP	1.9	1.7	1.2	2.8	2.5	1.8
GBM	0.9	0.8	0.7	1.2	0.9	1.1
NN	1.7	1.5	1.3	2.4	2.4	1.7

Table 6: Conditional root mean squared error of prediction for each LoB together with the process variance and estimation error. Over-dispersion parameters for the claim counts ϕ^N and claim amounts ϕ^X for each LoB and model.

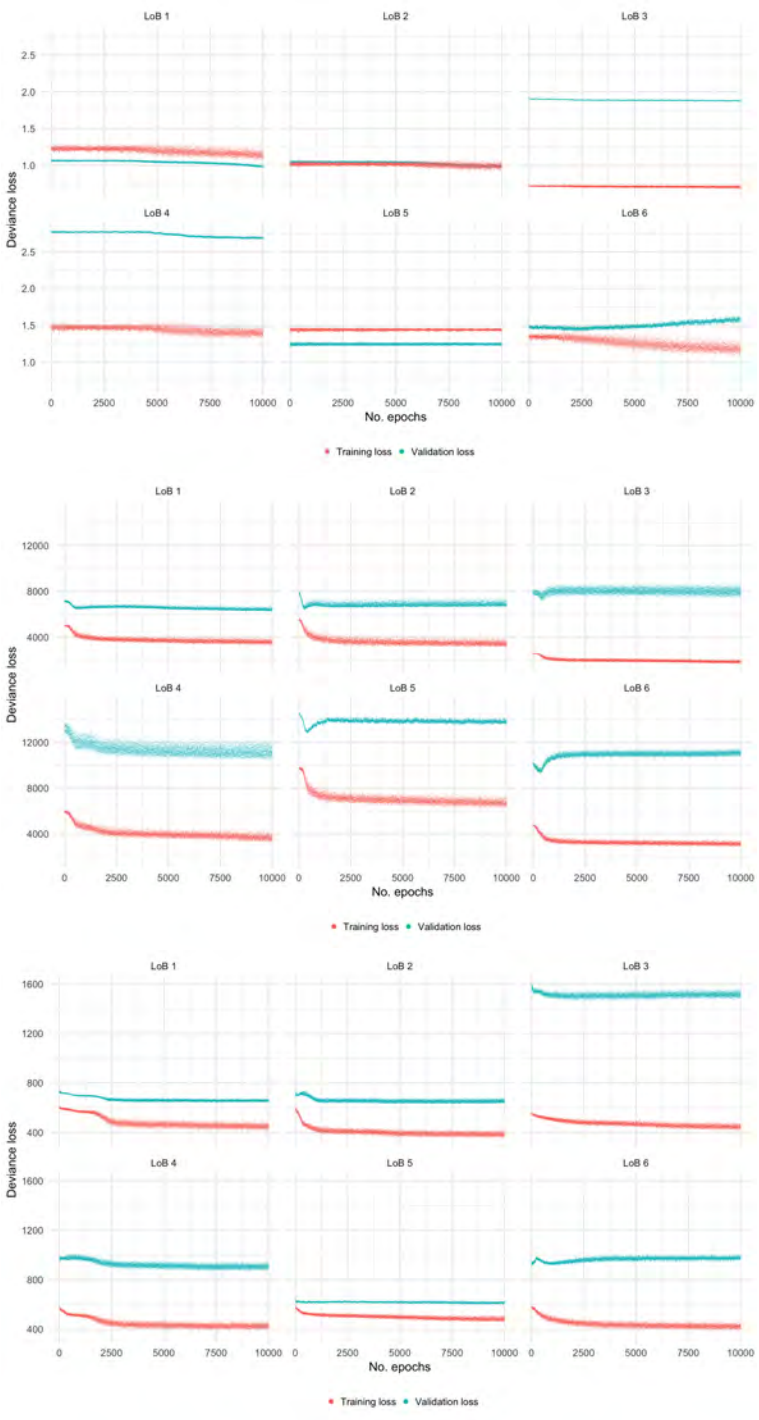

Figure 8: Training and validation error for the neural network models for each LoB. *Top:* Simple neural network claim counts part. *Middle:* Simple neural network claim amounts part. *Bottom:* Double neural network.

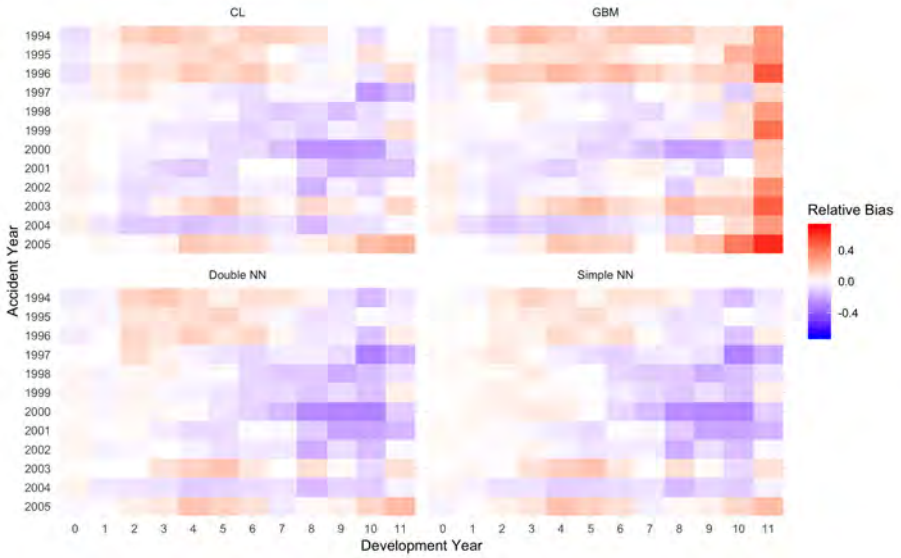

Figure 9: Heat maps of the relative biases for the reserving models for claim amounts for LoB 1.

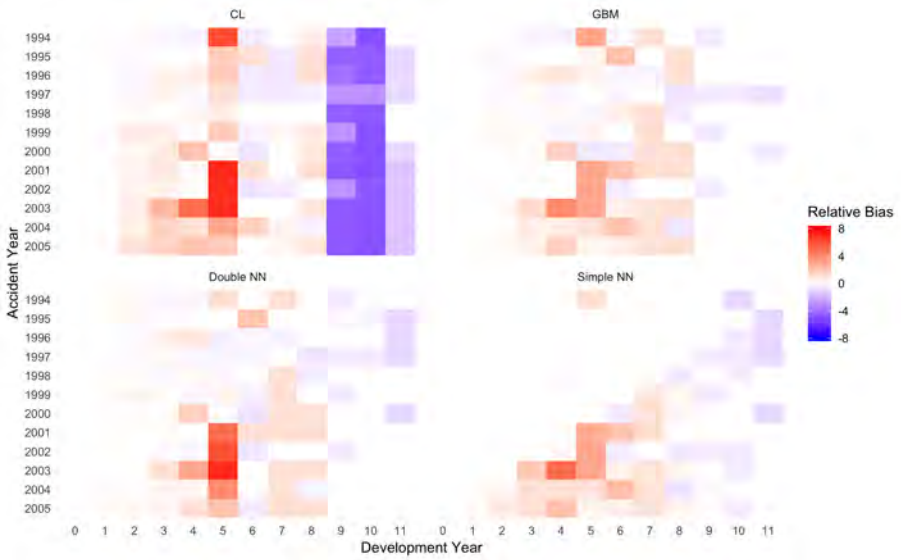

Figure 10: Heat maps of the relative biases for the reserving models for claim counts for LoB 1.

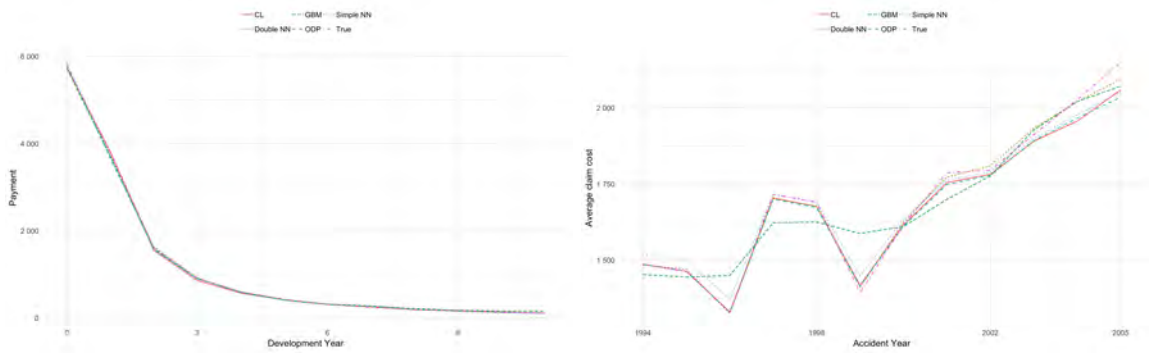

Figure 11: *Left:* True and predicted average payments per development year for LoB 6 (in thousands). *Right:* True and predicted average claim costs over accident years for LoB 6.

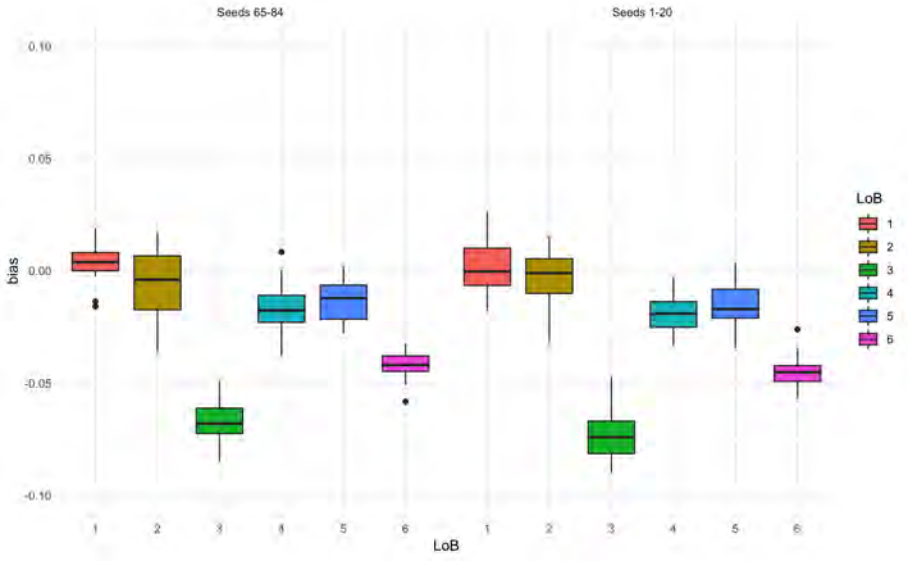

Figure 12: Boxplot of relative biases from predictions of double neural network model with different seeds.*Left:* Seeds 65-84. *Right:* Seeds 1-20.

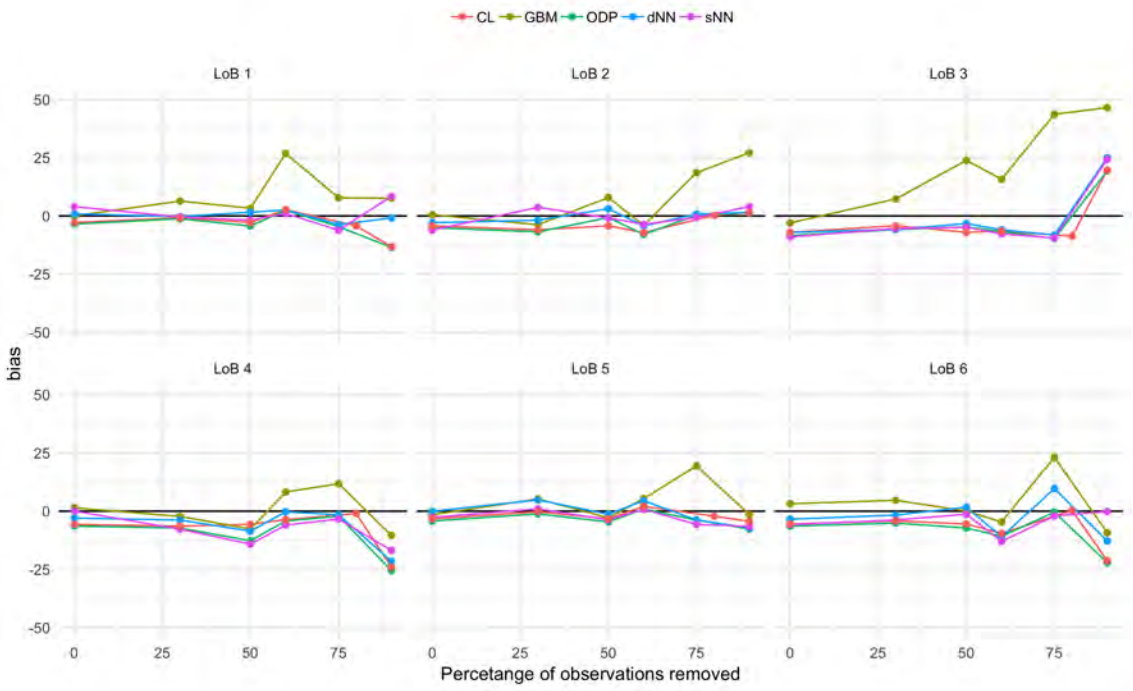

Figure 13: Relative bias for each LoB and model when removing part of the observations.

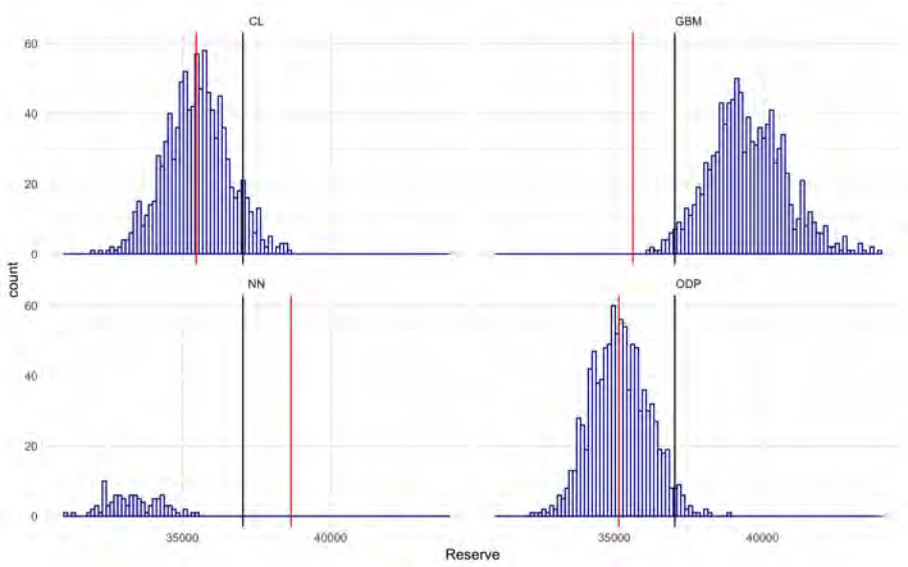

Figure 14: Histogram of the predicted reserves with parametric bootstrap for 1 000 simulations for LoB 2. For neural network only 100 simulations were performed. The black line shows the true reserve and the red line the predicted reserve as in Table 3.

LoB	N	Payment	RBNS %	Reserve
1	146 843	4 225 281	96.93	734 200
2	1 148 018	8 140 364	98.93	135 241
3	346 126	6 734 117	99.47	486 714

Table 7: Number of claims, total payments, percentage of the claims that are RBNS and outstanding reserve for each LoB (in thousands).

4 Application to real-world data

In this Section we apply the models discussed in Section 2 to real-world data and test their prediction ability.

4.1 Data Description

The data is from Folksam Ömsesidig Sakförsäkring and it consists of individual non-life insurance claims histories from three different LoBs. Similarly as in Section 3, we use three variables in our analysis: accident year, reporting delay and payment delay after reporting the claim. Some of the aggregated payments in the accident year, reporting and payment delay groups are negative and as in Section 3 we set the negative aggregated payments to zero. For LoB 1 0.57 % of the total payments are negative, for LoB 2 0.73 % and for Lob 3 0.13 %.

Table 7 shows the total number of claims, total claim payments, the percentage of the payments that are RBNS and the outstanding reserves for each LoB. The most claims are found in LoB 2 with more than 1 million claims and the fewest LoB 1 with 146 843 claims. The total payments are between 4 billion to 8 billion and the outstanding reserves varies between 135 million to 734 million. LoB 3 has the highest percentage of RBNS claims as 99.47 % of the claims are RBNS and LoB 2 has fewest with 96.93 % RBNS claims.

The average payment per development year for each LoB is shown in Figure 15 (right). The payments for LoB 1 start often after two years from the claim date as there are hardly any payments during the same year as the accident year. The claim payments peak after two years from the accident whereafter they decline and the claims are fully developed after 15 years.

The green line in Figure 15 shows the average payments per development year for LoB 2. Compared to LoB 1 the payment structure is quite different as most payments are made during the accident year. Finally, LoB 3 has quite similar development of payments as LoB 2. Here as well, most of the claim payments are made during the first two development year as we can see from the blue line in Figure 15 and are fully developed after 10 years.

Figure 15 illustrates the average claim cost for the three LoBs over the accident years. The average claim cost for LoB 1 was quite stable during the first eight years and started to decline after 1997. For LoB 2 there is a clear positive trend instead as the average claim

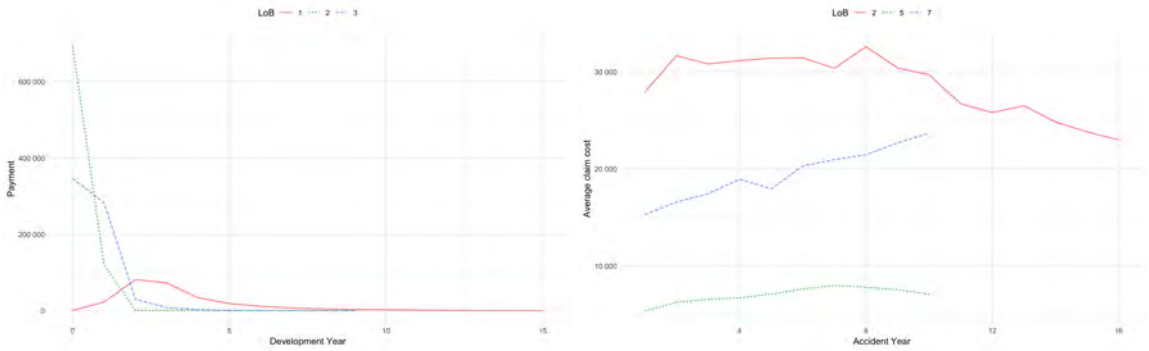

Figure 15: *Left:* Observed average payment per development year (in thousands). *Right:* Observed average claim cost over accident years.

cost have been rising since 1998 which is due to inflation. As we can see from the green line in Figure 15 the average claim cost for LoB 3 had a positive trend during the first seven years and thereafter we can notice a slight decrease in the claim costs. Thus, the highest claim costs are for claims from LoB 1 and the lowest from LoB 2.

4.2 Results

We perform the same model fitting procedure as in Section 3.2. The training and validation loss plots are very similar to Figures 4 - 8 and hence without showing the training and validation plots for GBM we observe that for all of these data sets the best fit is found with the same hyper-parameters as in Section 3.2, i.e. we choose shrinkage factor 0.01, bagging factor 1, i.e. no bagging, minimum observations per node 1 and interaction depth 2 for number of claims and 1 for claim amounts, i.e. no interactions. The number of trees used in fitting the claim counts and claim amounts models are summarised in Table 8. The claim counts model requires more trees compared to the claim amounts part and overall our real data set needs more trees compared to the simulated data, see Table 2. The validation loss in the neural network models have quite slow convergence and all of the LoBs need at least 6 000 epochs.

After finding the optimal tuning parameters, we fit the models for the upper left triangle and predict on the lower right triangle to obtain the predicted reserves. Table 9 shows true and predicted reserves for all the models and their relative biases. LoB 1 and 3 are more challenging compared to LoB 2. With GBM the relative bias for LoB 1 is only 6.77 % and -18.90 % for LoB 3. However, for LoB 2 the GBM model predicts slightly worse than the other models with relative bias 7.38 %. GBM succeeds to have the most accurate prediction of the outstanding reserves for LoB 1 as the other models have biases from -20 % to -45 %. The simple neural network model predicts the outstanding reserve for LoB 2 best as the relative bias is only -0.15 %.

		LoB		
Model	Type	1	2	3
GBM	Count	3 602	4 903	1 008
GBM	Amount	462	4 000	520
Simple NN	Count	7 227	6 308	8 729
Simple NN	Amount	9 721	8 629	6 526
Double NN	Claims	3 000	9 565	1 207

Table 8: Number of trees and epochs used in the machine learning models.

		LoB		
Model	Type	1	2	3
True	Reserve	734 200	135 241	486 714
CL	Reserve	401 572	131 799	375 972
CL	Bias %	-45.30	-2.55	-22.75
ODP	Reserve	459 873	141 439	374 627
ODP	Bias %	-37.36	4.58	-23.03
GBM	Reserve	783 878	145 219	394 719
GBM	Bias %	**6.77**	7.38	**-18.90**
Simple NN	Reserve	593 404	135 031	388 580
Simple NN	Bias %	-19.18	**-0.15**	-20.16
Double NN	Reserve	474 484	138 474	375 805
Double NN	Bias %	-35.37	2.39	-22.78

Table 9: True reserves and predicted reserves with relative biases for each LoB. The minimum relative bias for each LoB is in bold.

The simple neural network model here is fitted with trainable weights as the results are much better than with non-trainable weights, especially for LoB 1. The results with non-trainable weights were close to the ones we obtain with double neural network model. We tried also to fit the double neural network model with trainable input weights but this did not have significant effect on the predictions. The simple neural network model succeeds indeed to calibrate the ODP predictions to be more accurate. The same effect is not visible with the double neural network though where the predicted reserves are very close to the ones from the ODP reserving model. As in Section 3.2 we fitted the models when flipping the training and validation data but the predictions were very close to the predictions presented in Table 9.

The true and predicted IBNR and RBNS reserves are found in Table 10. The GBM model had the best predicted total reserves for LoB 1 but the relative bias for the IBNR reserve is terrible 99.80 %. With simulated data the IBNR biases were often worse than the RBNS biases (see Table 5) but here instead the IBNR biases in some cases are even

better than RBNS. For example GBM have relative bias for RBNS reserve of 15.56 % for LoB 2 and only -8.23 % for the IBNR reserve. Moreover, the double neural network model predicts IBNR reserves better than the RBNS reserves.

		LoB		
Model	Type	1	2	3
True	IBNR	114 617	46 497	30 311
ODP	IBNR	79 027	42 456	25 106
ODP	Bias %	-31.05	-8.69	-17.17
GBM	IBNR	229 007	42 668	24 708
GBM	Bias %	99.80	**-8.23**	-18.49
Simple NN	IBNR	80 601	40 507	24 538
Simple NN	Bias %	-29.68	-12.88	-19.04
Double NN	IBNR	85 551	41 103	25 101
Double NN	Bias %	**-25.36**	-11.60	**-17.18**
True	RBNS	619 583	88 743	456 403
ODP	RBNS	380 847	98 983	349 521
ODP	Bias %	-38.53	11.54	-23.42
GBM	RBNS	554 871	102 551	370 012
GBM	Bias %	**-10.44**	15.56	**-18.93**
Simple NN	RBNS	512 803	94 524	364 042
Simple NN	Bias %	-17.23	**6.51**	-20.24
Double NN	RBNS	388 933	97 370	350 104
Double NN	Bias %	-37.23	9.72	-23.29

Table 10: True RBNS and IBNR reserves and the predicted reserves with relative biases for each LoB.

We proceed with the predictions of the claim counts. The number of claims are showed in Table 11. We are mainly interested in the prediction ability of the IBNR claims since the models use the occurred claim counts when modelling the RBNS reserves, except the double neural network model where the number of claims are used for modelling both the RBNS and IBNR reserves. Notice that the bias for the RBNS claim counts is very small. GBM shows good performance here when it comes to the number of claims for LoB 3 as the bias is only -13 (-0.71 %). The neural network models instead seems to make even worse predictions than the ODP reserving model which is the starting point for the neural network models. This indicates that the neural network models do not calibrate the estimates from the ODP model very well.

Heat maps of the relative biases for the claim amounts for LoB 1 is shown in Figure 16. The worst predictions are found for accident year 9 and development year 15. In this cell the observed claim payment is much smaller than in the same development year for the other

		LoB		
Model	Type	1	2	3
True	IBNR	4 506	12 300	1 835
True	RBNS	133 665	1 007 143	314 079
ODP	IBNR	6 787	9 639	1 789
ODP	Bias %	50.62	-21.63	-2.51
GBM	IBNR	5 605	10 058	1 822
GBM	Bias %	**24.39**	**-18.23**	**-0.71**
Simple NN	IBNR	6 910	9 189	1747
Simple NN	Bias %	53.35	-25.29	-4.80
Double NN	IBNR	7 763	9 398	1 741
Double NN	Bias %	72.28	-23.59	-5.12
Double NN	RBNS	133 436	1 086 563	314 113
Double NN	Bias %	-0.17	0.87	0.01

Table 11: True claim counts and predicted claim counts with relative biases for each LoB for the lower right triangle.

accident years. GBM have the most extreme relative biases and yet it performs the best in total. Moreover, the machine learning models have more difficulties in predicting the claim payments for the first development year than Chain-Ladder as all of them overestimates the payments considerably. The relative biases for the claim counts are illustrated in Figure 17. The worst relative bias is for double neural network model for accident year 16 and development year 9. It is clear that the machine learning models predicts the number of claim better than Chain-Ladder. Observe that the results for the simple neural network model are rather similar to GBM.

In Figure 18 the true and predicted average payments per development year and average claim costs per accident year are illustrated. Although GBM predicts the outstanding reserves considerably better than the other models it does not capture the true structure for the average payments. Chain-Ladder and simple neural network have similar total predictions and they follow the true payment structure over the development years better. In Figure 18 we see that all the models underestimates the average claim cost severely for the two latest accident year which is reasonable as the claim cost during the first development years are considerably smaller than for the older years but the total payments after 15 years are still close to 200 000 as for the older years, see Table 18. Furthermore, GBM overestimates the average claim cost with more than 10 000 for accident year 11. To conclude, GBM have most bias per accident and development year but in total they weigh out which leads to a good total prediction.

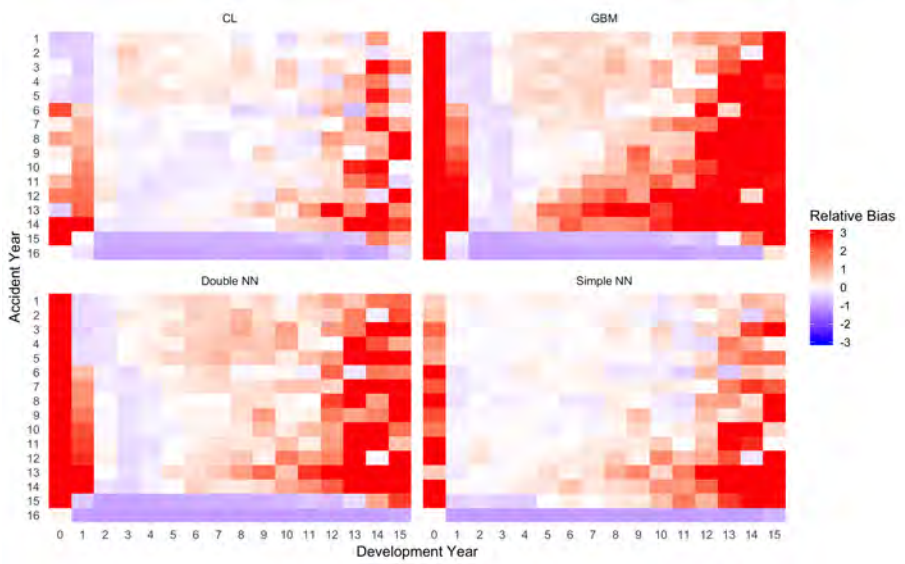

Figure 16: Heat maps of the relative biases for the reserving models for claim amounts for LoB 1. All biases over 300 % are set to 300 %.

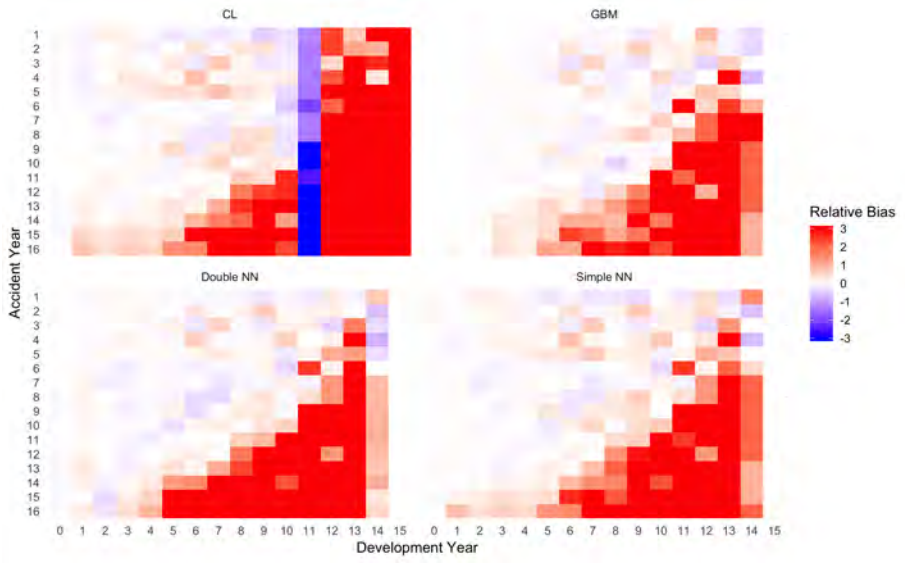

Figure 17: Heat maps of the relative biases for the reserving models for claim counts for LoB 1. All biases over 300 % and under -300 % are set to 300 % and -300 % respectively.

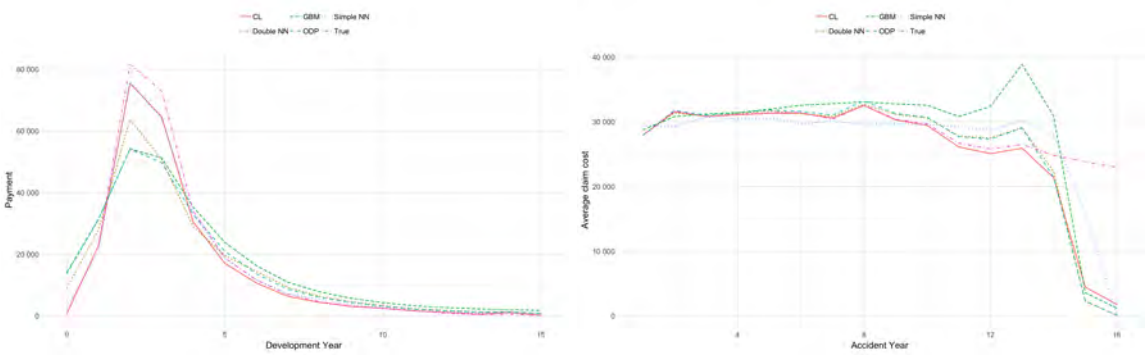

Figure 18: *Left:* True and predicted average payments per development year for LoB 1 (in thousands). *Right:* True and predicted average claim costs over accident years for LoB 1.

5 Discussion & Conclusions

In this thesis we have used both classical and machine learning methods for claims reserving. The goal has been to illustrate how machine learning models can be implemented in the claims reserving context.

What we have seen in Sections 3.2 and 4.2 is that the machine learning methods discussed in this thesis predict the outstanding reserves at least as well as the traditional Chain-Ladder model and often considerably better. This gives rise to use these methods in practice for example as a complement to Chain-Ladder. As we saw in Section 4 the real-world claims data might be more complicated to predict than the simulated data sets and in this case it would be helpful to use the machine learning methods in the analysis. Especially reserving for LoB 1 could be improved significantly with GBM or the simple neural network reserving models. To conclude, most often GBM had the best point estimate of the total outstanding reserves and hence we would choose this reserving model if the focus is on the best point estimate.

An advantage with the neural network models is that they are able to model non-linear dependencies of the covariates indirectly whereas with linear models one have to add these kind of dependencies explicitly. As we saw in Section 3.2, the double neural network model performed better than the simple in most cases. Thus, we can conclude that there is a non-linear relationship between the claim amounts and claim counts that the simple neural network model does not catch. Although, a disadvantage is that the neural network reserving model is more difficult to interpret than linear models.

The implementation of GBM in `R` is very straightforward and it requires less computation time than the neural network models. When implementing a neural network one has to define the input values for the covariates and set up the network architecture which requires more time from the practitioner. The handling of the covariates in the GBM reserving model is not predefined. Here, we have chosen to handle the covariates in the GBM model as numeric which allows for simple addition of the inflation effect. Another choice could be to treat the covariates as categorical as in ODP and neural network models. Notice that for the neural network models the categorical input covariates are crucial for the embedding of the ODP reserving model. Thus, we would recommend GBM if the simplicity of the model implementation is crucial for the choice of the reserving model.

If the goal is to choose a model that minimises the MSEP then based on the results in Table 6 CL or ODP would be the choice. These models would also be the choice is the goal is to avoid large estimation errors. As the machine learning models have more parameters compared to CL and ODP we do expect higher estimation error. Moreover, as we use parametric bootstrap and assume that the predicted values are the true ones we do not catch the model error. Thus, a model with large relative bias may still achieve low MSEP. Although, the neural network reserving model has many parameters it still has fairly low estimation error. Consequently, we would recommend neural network model over GBM if one seeks for a model with low estimation error.

As we saw in Section 3.2, the neural network models are quite sensitive for the choice of seed and division to training and validation data. Hence, in order to get a more stable prediction it could be beneficial to fit the model for a number of seeds and take the average prediction as the predicted outstanding reserve. Alternatively, one could use k-fold cross-validation for example by flipping the training and validation data and take the average of the predicted reserves.

In this thesis we have mostly included all the observation in the training data. The split of the training and validation data could also be chosen differently for example one could remove some of the observations for the older accident years and thus putting less weight on the older years and more on the latest years. This method could be especially beneficial for LoB 3 in the real-world data part where we observed that the average claim cost increased which probably caused that all of the methods underestimated the outstanding reserves with approximately 20 %.

The only tuning parameter for neural networks that we have considered here is the number of epochs as we have closely followed the architecture of the network from [4] and [5]. One could also optimise the number of neurons, dropout rate or activation functions for each hidden layer. This can be done following the same steps as we saw for GBM, i.e. comparing the training and validation losses for the different choices of tuning parameters. By optimising the tuning parameters the prediction accuracy might be improved for the neural network reserving models.

Notice that division with the overdispersion parameter in the double neural network model is necessary to regulate the difference of the volumes of the estimates, i.e. the deviances from the claim counts part are in general considerably smaller than the deviances from claim amounts part. This risks that the loss function in (73) focuses more on minimizing the loss from the claim amounts part and hence the claim amounts part is penalized more than the claim counts. Both here and when calculating MSEP we have used the number of parameters from the ODP reserving model to compute the overdispersion parameter but the actual number of parameters in the neural network models is noticeably larger. Moreover, in our case the number of parameters in the neural network models is even larger than the number of observations which implies that the estimated overdispersion parameter cannot be computed according to Pearson statistics. The estimation of the overdispersion parameter when considering the actual number of parameters in the model is left for future work.

6 References

[1] Bahdanau, D., Cho, K. & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv* 1409.0473, Version of May 19, 2016.

[2] Duval, F. & Pigeon, M. (2019). Individual loss reserving using a gradient boosting-based approach. *Risks*, 7(3):79.

[3] Efron, B. & Hastie, T. (2014). *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science.* Cambridge University Press.

[4] Gabrielli, A. (2020). A Neural Network Boosted Double Overdispersed Poisson Claims Reserving Model. *ASTIN Bulletin*, 50(1):25-60.

[5] Gabrielli, A., Richman, R. & Wüthrich, M. V. (2020). Neural Network Embedding of the Over-Dispersed Poisson Reserving Model. *Scandinavian Actuarial Journal*, 2020(1):1-29.

[6] Gabrielli, A. & Wüthrich, M. V. (2018). An Individual Claims History Simulation Machine.*Risks*, 6(2):29.

[7] Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press. Available at http://www.deeplearningbook.org.

[8] Hastie, T., Friedman, J. & Tibshirani, R. (2008). *The Elements of Statistical Learning*, Springer Series in Statistics.

[9] Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of Machine Learning Research*, 37:448-456.

[10] Lindholm, M., Verrall, R., Wahl, F. & Zakrisson, H. (2020). Machine Learning, Regression Models, and Prediction of Claims Reserves. *Casualty Actuarial Society E-Forum*, Summer.

[11] Luong, M.-T., Pham, H. and Manning, C.D. (2015). Effective Approaches to Attention-based Neural Machine Translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421.

[12] McCullagh, P. & Nelder, J. (1989). *Generalized Linear Models*. Chapman & Hall/CRC.

[13] Renshaw, A. & Verrall, R. (1998). A Stochastic Model Underlying the Chain-Ladder Technique. *British Actuarial Journal*, 4(4):903-923.

[14] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929-1958.

[15] Wüthrich, M. V. & Merz, M. (2019). Editorial: Yes, we CANN! *ASTIN Bulletin*, 49(1):1-3.

[16] Wüthrich, M. V. (2018). Machine learning in individual claims reserving. *Scandinavian Actuarial Journal*, 2018(6):465–480.

7 Appendix A

Tables 12 - 20 show the cumulative payments for simulated and real data sets used in Sections 3 - 4.

AY \ DY	0	1	2	3	4	5	6	7	8	9	10	11
1994	9 416	14 267	15 863	16 734	17 328	17 774	18 096	18 337	18 526	18 703	18 862	18 992
1995	9 822	15 115	16 941	17 967	18 624	19 081	19 445	19 737	19 965	20 156	20 302	20 440
1996	9 613	14 516	16 181	17 150	17 744	18 187	18 512	18 775	18 987	19 164	19 323	19 444
1997	9 788	15 038	16 861	17 946	18 691	19 241	19 672	19 975	20 201	20 397	20 608	20 775
1998	9 955	15 676	17 765	18 924	19 714	20 273	20 731	21 085	21 343	21 580	21 763	21 916
1999	10 453	16 575	18 789	20 100	20 959	21 589	22 086	22 442	22 721	22 945	23 138	23 285
2000	11 130	17 607	20 008	21 363	22 253	22 930	23 442	23 853	24 207	24 515	24 765	24 950
2001	11 268	17 896	20 401	21 894	22 901	23 589	24 073	24 435	24 736	25 020	25 251	25 453
2002	11 475	18 427	21 075	22 553	23 506	24 206	24 722	25 107	25 444	25 693	25 913	26 088
2003	12 172	19 256	22 002	23 398	24 252	24 834	25 313	25 694	25 955	26 191	26 405	26 571
2004	12 816	20 844	23 943	25 696	26 878	27 701	28 308	28 748	29 113	29 401	29 640	29 841
2005	13 239	20 648	23 398	24 896	25 770	26 414	26 918	27 342	27 640	27 886	28 073	28 233

Table 12: Cumulative claim payments for LoB 1 simulated data.

AY \ DY	0	1	2	3	4	5	6	7	8	9	10	11
1994	9 638	14 449	16 107	17 024	17 645	18 062	18 374	18 622	18 807	18 990	19 128	19 239
1995	10 042	15 035	16 807	17 774	18 407	18 853	19 185	19 439	19 642	19 824	19 977	20 102
1996	9 515	14 429	16 121	17 102	17 658	18088	18 406	18 657	18 848	19 020	19 177	19 285
1997	9 659	14 336	15 935	16 807	17 364	17 724	17 983	18 147	18 265	18 353	18 416	18 489
1998	10 298	15 948	18 051	19 270	20 197	20 844	21 319	21 683	21 965	22 223	22 427	22 589
1999	10 722	17 337	19 868	21 339	22 340	22 995	23 500	23 943	24 313	24 592	24 851	25 077
2000	10 363	16 250	18 339	19 599	20 384	20 950	21 381	21 680	21 908	22 109	22 278	22 404
2001	11 113	17 429	19 939	21 335	22 275	22 927	23 380	23 777	24 079	24 340	24 549	24 721
2002	11 316	17 729	20 066	21 393	22 220	22 758	23 139	23 438	23 657	23 824	23 988	24 138
2003	12 609	19 894	22 733	24 287	25 292	25 978	26 488	26 915	27 264	27 560	27 816	28 039
2004	12 387	19 751	22 510	24 019	25 013	25 681	26 194	26 548	26 834	27 089	27 311	27 525
2005	12 798	19 867	22 558	24 002	24 935	25 510	25 936	26 290	26 547	26 724	26 886	27 014

Table 13: Cumulative claim payments for LoB 2 simulated data.

AY \ DY	0	1	2	3	4	5	6	7	8	9	10	11
1994	2 888	4 382	4 881	5 181	5 370	5 497	5 584	5 626	5 657	5 697	5 742	5 770
1995	3 098	4 697	5 280	5 602	5 811	5 969	6 100	6 196	6 263	6 328	6 384	6 426
1996	3 125	4 710	5 258	5 578	5 803	5 971	6 095	6 195	6 290	6 348	6 387	6 419
1997	3 411	5 451	6 157	6 580	6 831	6 989	7 123	7 241	7 307	7 381	7 439	7 497
1998	3 564	5 407	5 979	6 264	6 466	6 612	6 735	6 827	6 893	6 951	6 987	7 027
1999	3 780	5 772	6 457	6 831	7 119	7 326	7 463	7 573	7 644	7 715	7 784	7 838
2000	3 996	6 197	6 937	7 374	7 651	7 862	7 992	8 097	8 204	8 281	8 363	8 425
2001	4 800	7 583	8 564	9 122	9 483	9 748	9 926	10 079	10 196	10 292	10 381	10 457
2002	4 686	7 273	8 268	8 787	9 092	9 315	9 479	9 667	9 776	9890	9 955	10 021
2003	5 380	8 555	9 752	10 369	10 776	11 068	11 257	11 438	11 564	11 660	11 738	11 805
2004	5 890	9 321	10 644	11 487	11 973	12 303	12 507	12 664	12 773	12 934	13 020	13 098
2005	6 163	9 789	11 192	11 941	12 394	12 720	12970	13 144	13 273	13 379	13 465	13 560

Table 14: Cumulative claim payments for LoB 3 simulated data.

AY \ DY	0	1	2	3	4	5	6	7	8	9	10	11
1994	12 442	19 854	22 590	24 162	25 222	25 950	26 533	27 007	27 399	27 730	28 013	28 246
1995	11 986	19 094	21 714	23 233	24 248	24 960	25 470	25 861	26 176	26 460	26 728	26 940
1996	12 404	19 994	22 887	24 530	25 620	26 451	27 047	27 575	27 997	28 331	28 618	28 844
1997	12 508	20 004	22 910	24 646	25 805	26 605	27 194	27 639	27 972	28 289	28 576	28 792
1998	14 002	23 436	27 563	29 960	31 583	32 726	33 595	34 340	34 901	35 415	35 797	36 126
1999	13 031	22 306	25 976	28 086	29 542	30 586	31 422	32 108	32 667	33 176	33 606	33 956
2000	14 097	23 707	27 594	29 930	31 525	32 700	33 541	34 216	34 746	35 174	35 554	35 868
2001	15 052	25 770	30 394	33 026	34 937	36 263	37 317	38 243	38 949	39 553	40 063	40 441
2002	15 812	25 937	30 138	32 485	34 050	35 151	36 012	36 678	37 186	37 606	37 963	38 266
2003	16 469	28 397	33 375	36 203	38 197	39 638	40 700	41 531	42 204	42 755	43 229	43 679
2004	17 341	29 320	34 467	37 472	39 465	40 834	41 866	42 673	43 210	43 660	44 120	44 441
2005	17 856	29 761	34 617	37 410	39 292	40 543	41 502	42 175	42 669	43 035	43 435	43 746

Table 15: Cumulative claim payments for LoB 4 simulated data.

AY \ DY	0	1	2	3	4	5	6	7	8	9	10	11
1994	12 195	19 207	21 725	23 159	24 208	24 951	25 512	25 950	26 286	26 577	26 818	27 011
1995	12 929	20 818	23 907	25 673	26 899	27 835	28 533	29 024	29 478	29 832	30 174	30 450
1996	12 720	20 953	24 263	26 163	27 427	28 416	29 179	29 760	30 220	30 643	31 003	31 295
1997	13 833	22 338	25 775	27 723	29 035	30 021	30 789	31 332	31 798	32 229	32 588	32 869
1998	14 131	23 945	28 096	30 541	32 259	33 516	34 465	35 262	35 875	36 404	36 885	37 282
1999	13 072	21 489	24 929	26 916	28 278	29 269	30 018	30 565	31 018	31 375	31 723	31 996
2000	14 703	24 834	29 006	31 419	33 088	34 320	35 331	36 117	36 787	37 364	37 894	38 355
2001	14 825	23 994	27 815	30 093	31 616	32 648	33 301	33 783	34 196	34 562	34 863	35 059
2002	15 514	26 368	31 046	33 723	35 555	36 818	37 821	38 582	39 125	39 614	40 005	40 338
2003	16 045	27 178	31 866	34 635	36 413	37 677	38 631	39 311	39 870	40 320	40 681	41 055
2004	17 480	30 213	36 115	39 525	41 835	43 421	44 580	45 466	46 119	46 613	47 078	47 388
2005	18 536	30 523	35 359	38 117	39 882	41 155	42 097	42 821	43 398	43 859	44 269	44 628

Table 16: Cumulative claim payments for LoB 5 simulated data.

AY \ DY	0	1	2	3	4	5	6	7	8	9	10	11
1994	4 645	7 539	8 616	9 213	9 626	9 926	10 159	10 344	10 483	10 622	10 727	10 813
1995	5 174	8 028	9 159	9 817	10 250	10 559	10 791	10 974	11 124	11 255	11 368	11 444
1996	4 500	7 096	8 026	8 557	8 934	9 236	9 465	9 643	9 785	9 913	10 004	10 083
1997	4 904	8 318	9 645	10 410	10 943	11 371	11 685	11 940	12 121	12 294	12 436	12 595
1998	4 891	8 272	9 720	10 536	11 077	11 477	11 790	12 042	12 246	12 438	12 587	12 700
1999	4 596	7 467	8 592	9 209	9 568	9 833	10 029	10 186	10 319	10 398	10 457	10 523
2000	5 247	8 782	10 279	11 100	11 677	12 073	12 375	12 594	12 799	12 967	13 098	13 201
2001	5 956	9 990	11 686	12 646	13 311	13 780	14 163	14 482	14 726	14 945	15 134	15 298
2002	6 387	10 671	12 522	13 614	14 315	14 850	15 225	15 501	15 737	15 924	16 080	16 203
2003	6 811	11 653	13 871	15 162	15 989	16 528	16 901	17 207	17 416	17 576	17 716	17 837
2004	7 431	12 822	15 278	16 702	17 615	18 236	18 596	18 856	19 088	19 295	19 439	19 559
2005	8 388	14 203	16 678	18 088	18 991	19 630	20 075	20 416	20 694	20 916	21 082	21 227

Table 17: Cumulative claim payments for LoB 6 simulated data.

AY \ DY	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1990	1936	43 294	120 841	172 234	196 610	210 598	219 812	226 035	231 294	234 392	238 104	239 580	240 461	241 262	241 765	242 074
1991	1676	48 853	147 859	192 838	220 941	239 590	249 603	256 548	260 317	263 986	267 139	268 966	270 294	270 866	272 728	27 3031
1992	1109	58 800	154 259	209 938	240 284	255 826	267 502	273 999	277 598	280 995	282 781	285 149	286 117	286 710	287 013	287 150
1993	1462	52 914	161 194	223 546	253 213	274 219	285 197	291 802	296 280	299 606	301 674	304 180	306 575	306 940	307 462	307 989
1994	1790	63 049	181 066	245 148	275 631	292 691	305 081	311 433	317 039	320 598	324 126	325 866	326 952	327 410	327 811	328 031
1995	323	18 061	102 030	190 459	220 657	241 487	253 734	260 531	268 731	273 717	277 210	280 474	281 209	283 275	283 851	284 259
1996	897	14 853	107 534	187 555	230 578	249 691	262 851	270 509	276 141	279 958	282 286	283 867	285 295	285 654	285 958	286 140
1997	525	15 598	94 300	189 305	235 196	257 279	275 155	286 450	291 897	296 101	299 818	302 760	303 446	303 868	304 627	304 726
1998	942	14 106	102 536	194 097	234 188	258 717	273 993	284 327	289 858	292 410	296 149	298 621	299 497	300 264	300 912	300 926
1999	1113	12 127	97 967	186 954	227 545	255 509	271 178	281 036	287 161	290 329	293 533	295 142	296 220	296 416	296 696	297 081
2000	512	9 360	74 406	153 962	199 983	222 410	236 929	244 149	249 617	253 360	255 434	257 545	258 457	258 690	259 023	259 569
2001	334	8 270	65 382	153 017	192 590	213 103	224 169	230 977	234 911	238 901	240 240	241 403	242 175	242 492	244 104	243 325
2002	1684	9 123	78 735	159 715	196 129	211 266	222 744	228 607	231 683	233 403	235 478	236 197	236 310	236 548	236 718	236 844
2003	132	4 483	73 944	137 904	165 127	180 414	187 364	193 002	196 884	199 342	200 614	201 385	201 870	201 980	202 059	202 126
2004	23	3 834	64 649	131 496	157 915	170 336	177 372	182 497	185 704	187 948	189 163	189 716	190 331	190 544	190 612	190 640
2006	63	2 208	57 076	123 388	149 283	164 484	171 104	173 678	175 640	177 255	178 787	180 128	180 474	180 896	181 333	181 369

Table 18: Cumulative claim payments for LoB 2 real data (in thousands).

AY \ DY	0	1	2	3	4	5	6	7	8	9
1998	227 336	316 071	313 741	313 994	314 398	314 174	314 336	314 358	314 370	314 392
1999	663 745	761 039	757 558	756 495	755 995	756 377	756 457	756 509	756 583	756 648
2000	739 064	843 048	836 355	823 719	836 903	836 963	836 925	837 029	837 050	837 188
2001	723 910	852 387	853 819	854 743	854 732	854 751	854 801	854 864	854 771	854 865
2002	743 203	870 406	873 217	874 318	875 153	875 308	875 373	875 514	875 529	875 586
2003	746 454	870 587	872 058	874 744	875 058	874 959	875 118	875 322	875 424	875 475
2004	799 155	940 223	943 321	944 389	945 071	945 169	945 287	945 488	945 531	945 468
2005	827 690	960 643	962 556	963 506	963 618	963 807	964 470	964 472	964 518	964 577
2006	747 098	882 656	883 038	884 360	885 292	885 691	885 768	885 780	885 698	885 695
2007	702 351	827 073	829 111	829 912	830 125	830 260	830 307	830 316	830 322	830 469

Table 19: Cumulative claim payments for LoB 5 real data (in thousands).

AY \ DY	0	1	2	3	4	5	6	7	8	9
1998	259 064	432 045	447 060	455 107	458 230	459 847	460 363	461 695	461 815	461 864
1999	286 494	545 344	569 480	576 245	578 451	575 951	576 446	576 867	577 453	577 573
2000	348 800	635 330	666 144	674 992	677 419	677 298	677 951	679 812	680 948	682 265
2001	369 085	662 471	696 755	705 912	708 861	710 564	711 129	712 228	712 989	713 324
2002	361 526	640 113	665 701	673 664	676 697	678 119	679 214	680 031	680 395	680 438
2003	368 598	641 813	667 759	669 925	671 551	673 122	673 551	673 923	674 173	674 218
2004	334 501	630 481	656 391	664 007	666 074	666 255	666 813	667 933	667 775	668 137
2005	446 834	758 596	796 982	804 656	810 007	812 864	811 236	811 557	811 669	811 779
2006	360 231	675 552	715 969	728 853	732 065	733 185	737 022	735 929	734 645	737 232
2007	330 935	671 189	714 700	722 895	726 676	725 816	726 782	726 840	727 104	727 287

Table 20: Cumulative claim payments for LoB 7 real data (in thousands).

8 Appendix B

In this section we show the training and validation plots for the GBM models for LoBs 2-6.

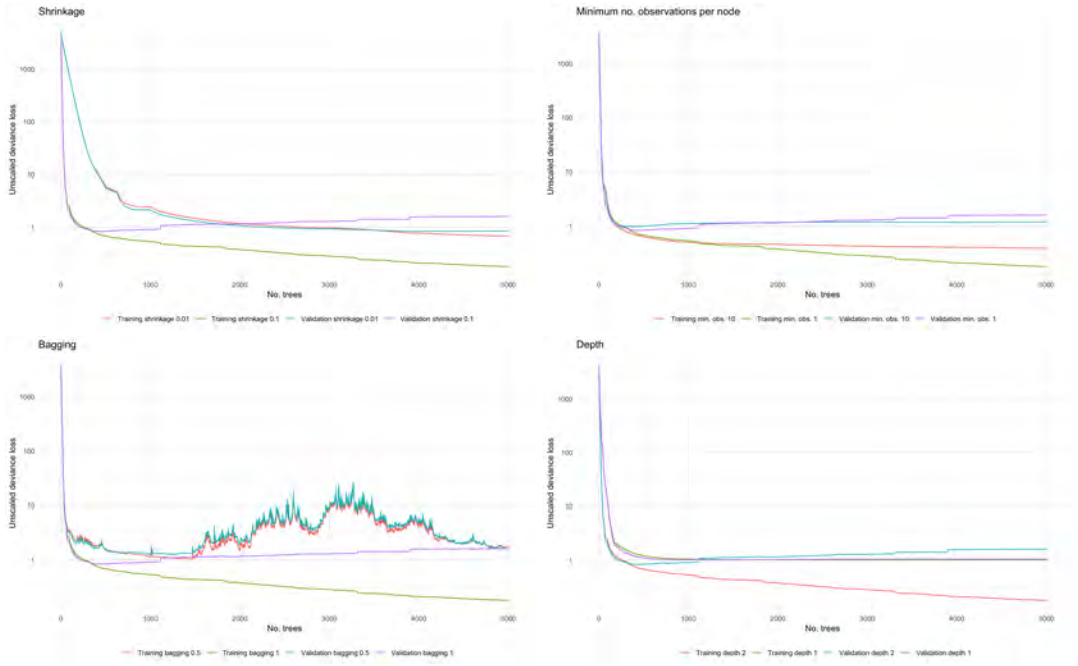

Figure 19: Training and validation error for the claim counts part of GBM for LoB 2. *Up left:* Shrinkage factor 0.1 and 0.01. *Up right:* Minimum observations per node 1 and 10. *Bottom left:* Bagging factor 0.5 and 1. *Bottom right:* Interaction depth 1 and 2.

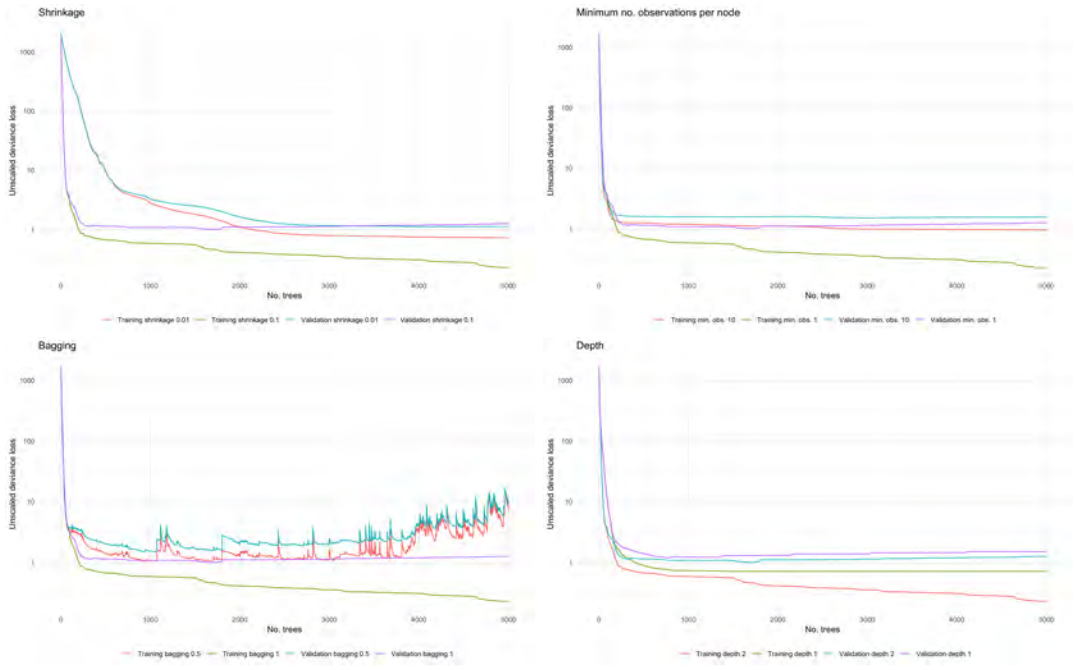

Figure 20: Training and validation error for the claim counts part of GBM for LoB 3. *Up left:* Shrinkage factor 0.1 and 0.01. *Up right:* Minimum observations per node 1 and 10. *Bottom left:* Bagging factor 0.5 and 1. *Bottom right:* Interaction depth 1 and 2.

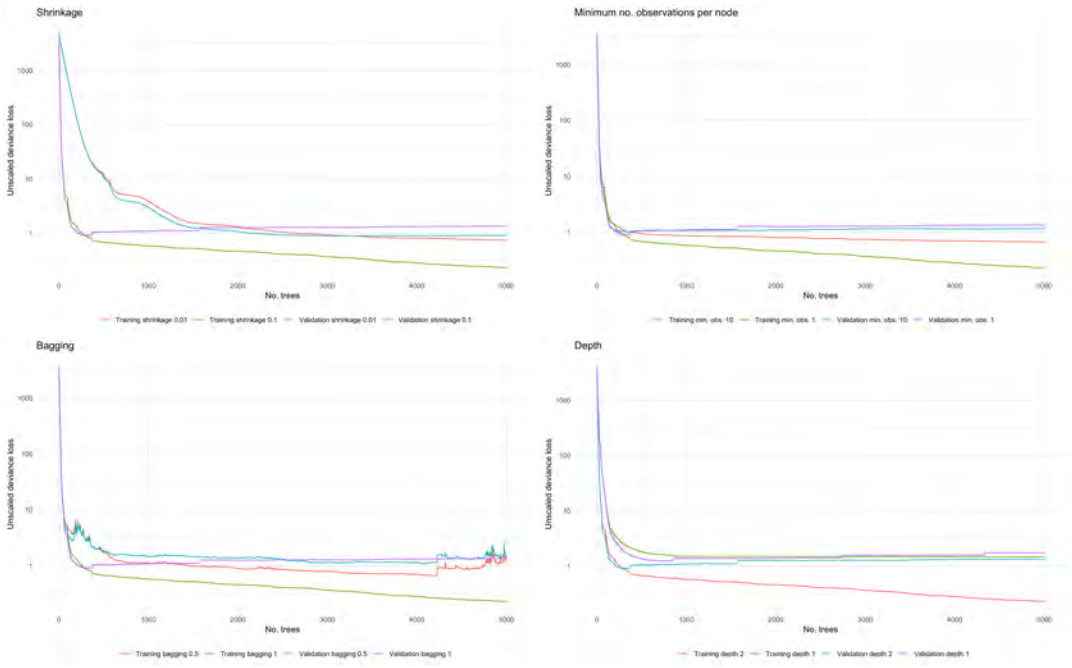

Figure 21: Training and validation error for the claim counts part of GBM for LoB 4. *Up left:* Shrinkage factor 0.1 and 0.01. *Up right:* Minimum observations per node 1 and 10. *Bottom left:* Bagging factor 0.5 and 1. *Bottom right:* Interaction depth 1 and 2.

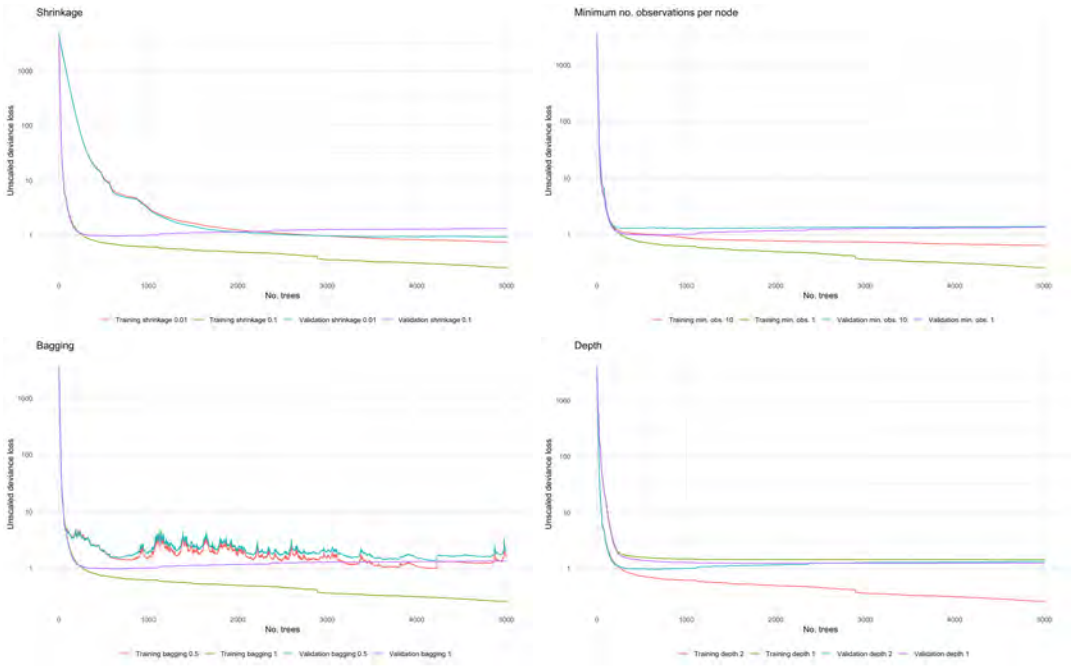

Figure 22: Training and validation error for the claim counts part of GBM for LoB 5. *Up left:* Shrinkage factor 0.1 and 0.01. *Up right:* Minimum observations per node 1 and 10. *Bottom left:* Bagging factor 0.5 and 1. *Bottom right:* Interaction depth 1 and 2.

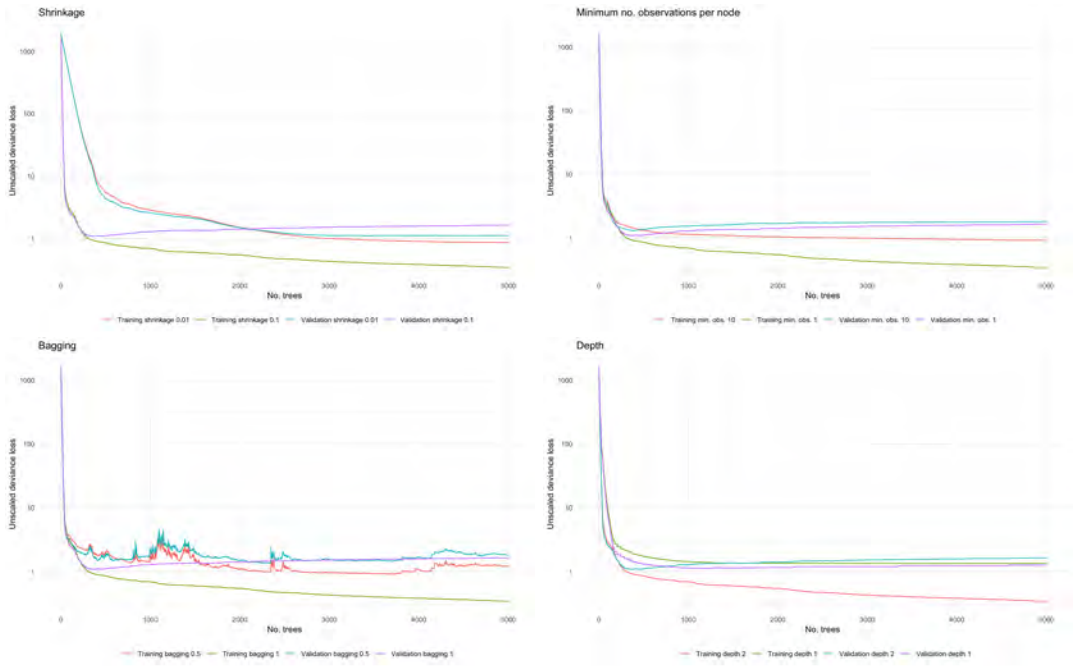

Figure 23: Training and validation error for the claim counts part of GBM for LoB 6. *Up left:* Shrinkage factor 0.1 and 0.01. *Up right:* Minimum observations per node 1 and 10. *Bottom left:* Bagging factor 0.5 and 1. *Bottom right:* Interaction depth 1 and 2.

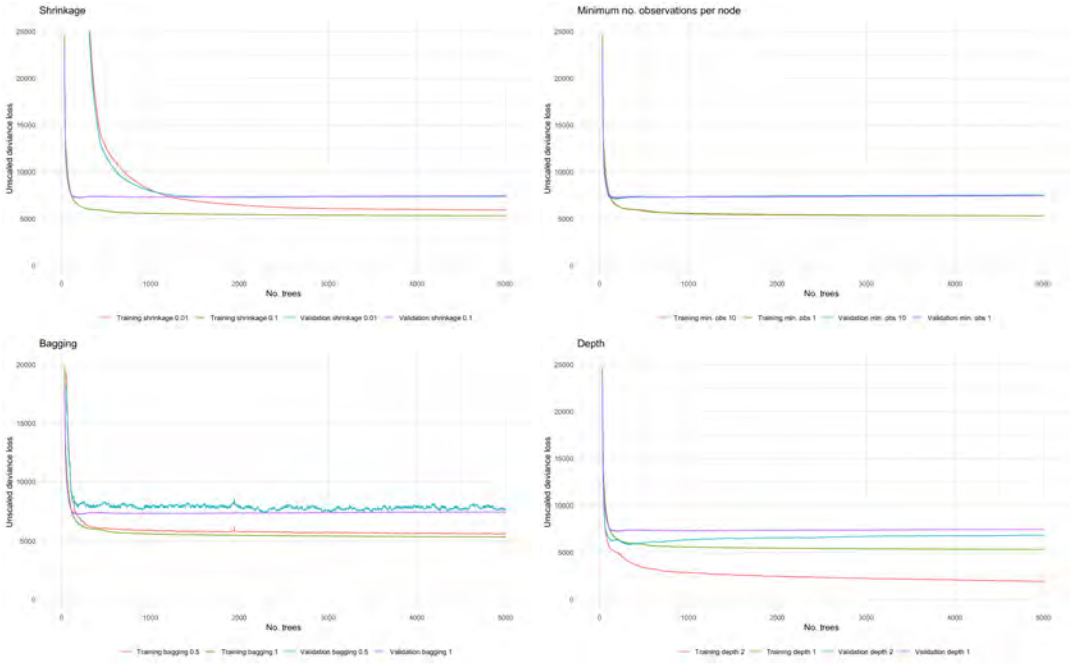

Figure 24: Training and validation error for the claim amounts part of GBM for LoB 2. *Up left:* Shrinkage factor 0.1 and 0.01. *Up right:* Minimum observations per node 1 and 10. *Bottom left:* Bagging factor 0.5 and 1. *Bottom right:* Interaction depth 1 and 2.

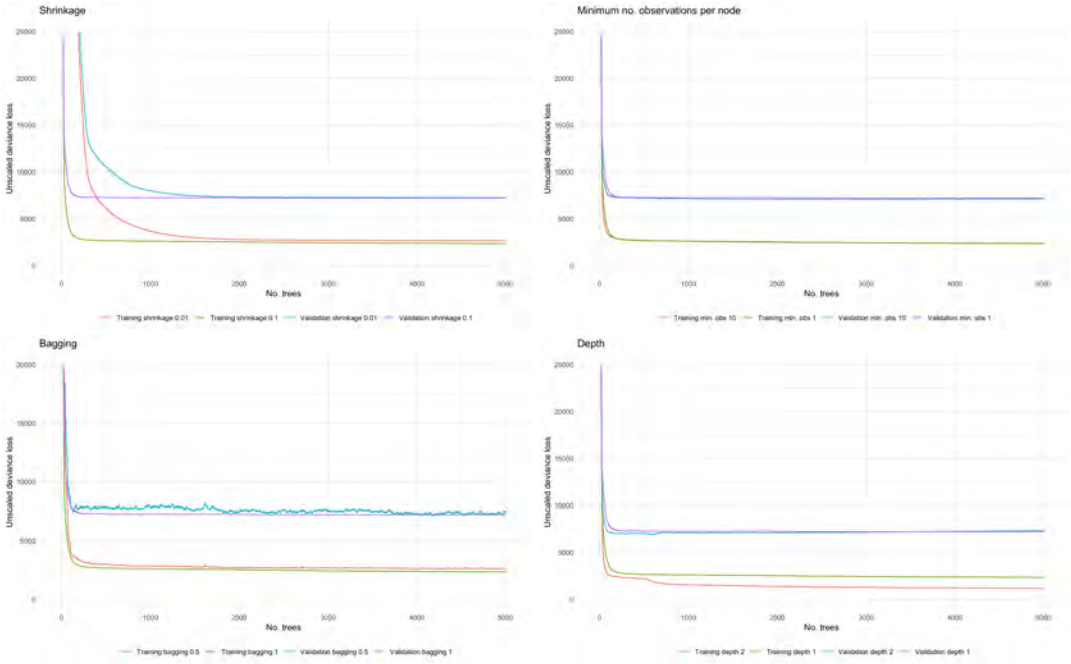

Figure 25: Training and validation error for the claim amounts part of GBM for LoB 3. *Up left:* Shrinkage factor 0.1 and 0.01. *Up right:* Minimum observations per node 1 and 10. *Bottom left:* Bagging factor 0.5 and 1. *Bottom right:* Interaction depth 1 and 2.

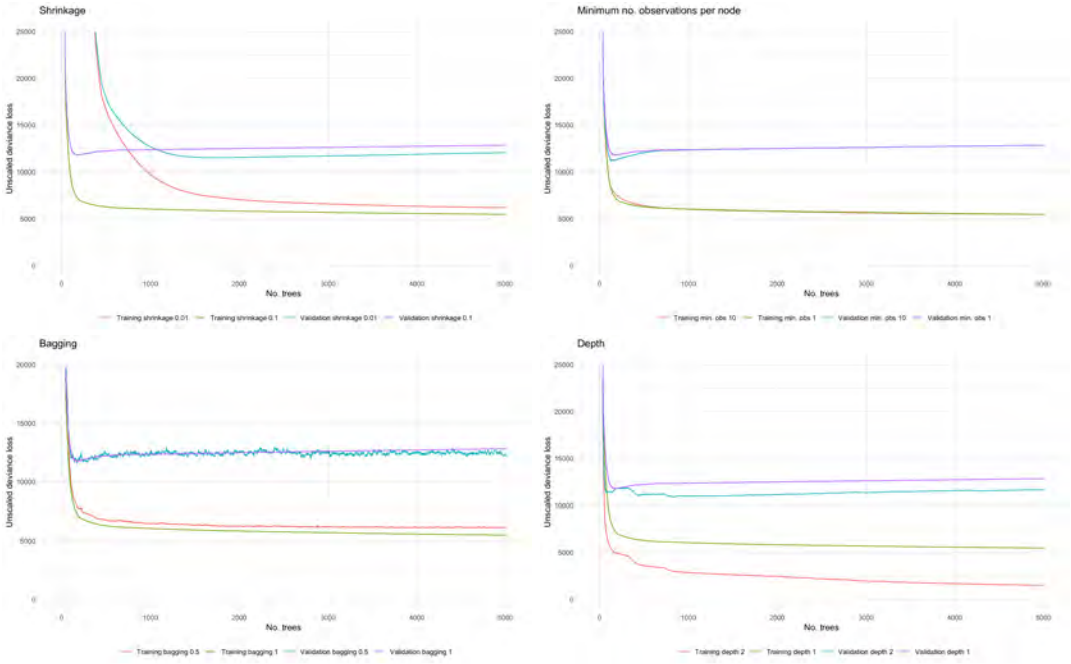

Figure 26: Training and validation error for the claim amounts part of GBM for LoB 4. *Up left:* Shrinkage factor 0.1 and 0.01. *Up right:* Minimum observations per node 1 and 10. *Bottom left:* Bagging factor 0.5 and 1. *Bottom right:* Interaction depth 1 and 2.

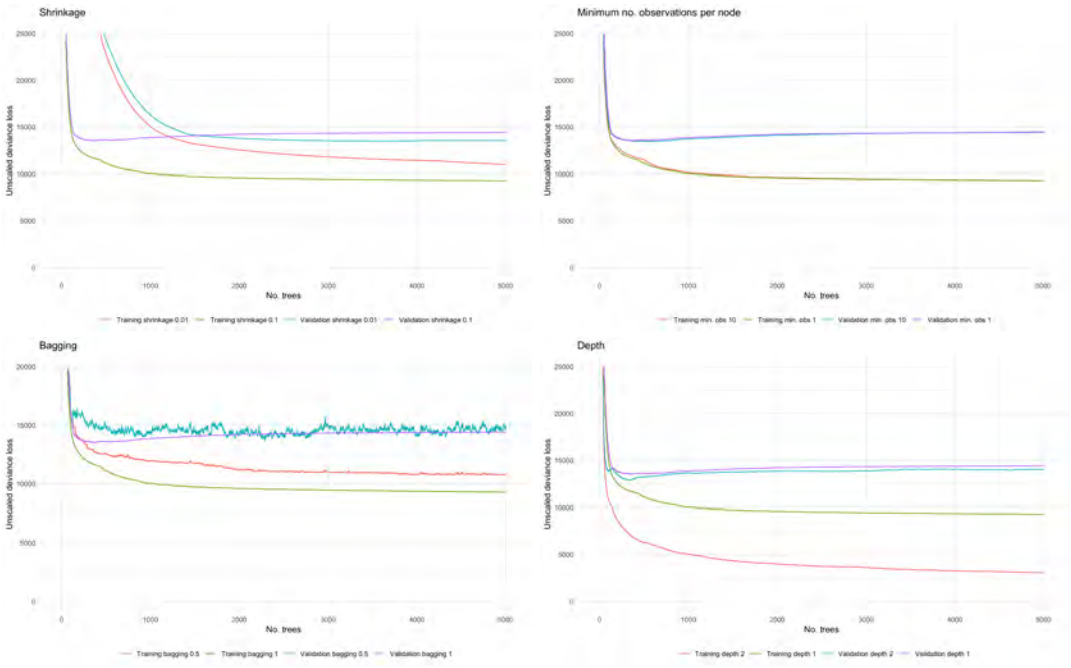

Figure 27: Training and validation error for the claim amounts part of GBM for LoB 5. *Up left:* Shrinkage factor 0.1 and 0.01. *Up right:* Minimum observations per node 1 and 10. *Bottom left:* Bagging factor 0.5 and 1. *Bottom right:* Interaction depth 1 and 2.

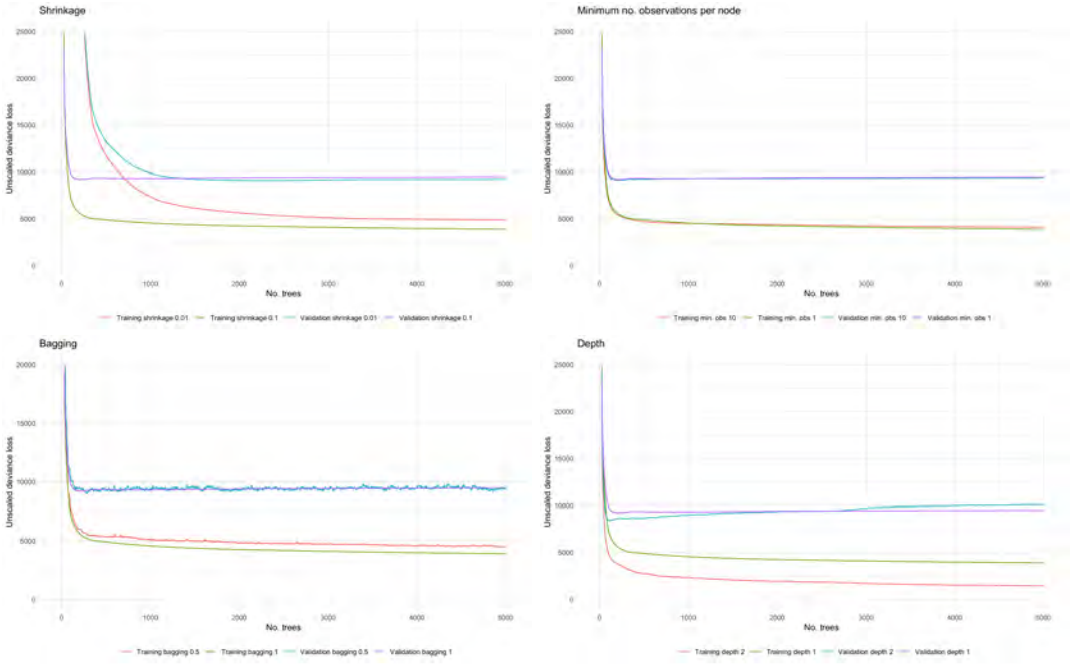

Figure 28: Training and validation error for the claim amounts part of GBM for LoB 6. *Up left:* Shrinkage factor 0.1 and 0.01. *Up right:* Minimum observations per node 1 and 10. *Bottom left:* Bagging factor 0.5 and 1. *Bottom right:* Interaction depth 1 and 2.