ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Master Thesis Quantitative Finance

Using decision tree ensemble methods for the estimation of individual claims reserving

Author:
Fan Fan Jin (449599)
External supervisor:
MSc. I.S. Nonneman

Academic supervisor:
Dr. A.A. Naghi
Second assessor:
Dr. M. Grith

June 29, 2021

Abstract

The aim of this research is to estimate individual claims reserving using the decision tree ensemble methods: XGBoost, Random Forest and Extra Trees, with the focus on the prediction of Reported But Not Settled (RBNS) claims. The results obtained from these three methods are compared with the traditional triangular method, such that we are able to conclude if using data on micro level and the use of machine learning methods can improve the predictions of individual claims reserving. Also, we use an interpretation method Tree SHAP to be able to interpret the results obtained from the decision tree ensemble methods. We find that the XGBoost algorithm outperforms the Random Forest and Extra Trees algorithm in its predictions for individual claims reserving. When comparing the results obtained from XGBoost with the traditional triangular method, we find that the estimates obtained from the traditional triangular method are closer to the actual reserves than the XGBoost algorithm. However, the differences are small and the advantage of using micro level data and machine learning methods is that it allows us to interpret the drivers of claims reserves. Using the Tree SHAP method we find that the age of the policyholder, the quarter of the year in which the accident took place, the number of development years between the training and test set and a specific injured body part are the most important variables for the prediction of individual claims reserving using all the three decision tree ensemble methods.

Contents

1 Introduction

Claims reserving is one of the most important tasks non-life insurers have to deal with. Due to regulations, such as the Solvency II Framework and increased requirements on financial reporting, insurers are obliged to have enough reserves to reduce the risk of insolvency. Insurers need to estimate future losses, in order to meet their future liabilities, such that they have enough funds to pay out in times of financial distress. A proper estimation of claims reserving will reduce the risk of insolvency and increase the insurer's financial well-being.

Traditional claims reserving methods, so-called claims reserving triangles (Mack (1993)), have been known for quite some time now. Most non-life insurers nowadays still use these traditional triangular methods. These methods use aggregated data to predict the claims reserving, also known for estimating claims reserves on macro level. This means that information per claim on micro level, such as age of the policyholder or type of claim, is not incorporated in the estimation process. The only information needed for the traditional triangular methods are the history of the claims amount paid per year.

In the past, insurers did not always have access to large data sets with specific information per claim. Therefore, they did not have any other choice than using these traditional triangular methods. However, in the last decade due to big-data processes being very booming in the industry, insurers have also been collecting more information per claim due to the right resources. With the advanced computing power over the years and the availability of large data sets, recent researchers have proposed methods to estimate the claims reserving on micro level (or: individual level) using specific claim data.

Data on micro level provides more detailed information, that can increase the interpretability of the estimation of claims reserving. The extra information used in micro level models could for example be explicit details of the policyholder or claim development information. Overall, incorporating this kind of information into reserving models helps the insurer (and the regulator) to get a better understanding into the insurance risks. Therefore, it is of interest for non-life insurers to delve into micro level models for their estimation of reserves instead of using models with aggregated data.

Past researchers have already been using individual claims data for the estimation of claims reserving. There are multiple methods and ways to do this, for example using a Generalized Linear Model (GLM) (Zhou and Garrido (2009), Crevecoeur and Antonio (2019b)), a Poisson process (Antonio and Plat (2010)), copulas (Zhao and Zhou (2010)), a multivariate skew-symmetric distribution (Pigeon et al. (2013b)) etc. Among these methods, machine learning techniques (Wüthrich (2018a), Wüthrich (2018b), Duval and Pigeon (2019), Baudry and Robert (2019)) are also very popular to use, because of their flexibility and their capability of using structured and unstructured information. Duval and Pigeon (2019) have stated in their points of future research that it might be interesting to compare results of different machine learning methods on the same data set.

There are different machine learning methods in the literature, such as neural networks (Wüthrich, 2018b), support vector machines (Ticconi, 2018) or methods using decision trees (Baudry and Robert (2019), Duval and Pigeon (2019)). Among these three methods, decision trees are the most interpretable. Therefore, our research focuses on decision trees, where we consider the decision tree ensemble methods, XGBoost, Random Forest and Extra Trees, with its (dis)advantages and compare the results of individual claims reserving obtained from these techniques. We will also compare this

with one of the traditional triangular methods, Mack Chain Ladder method (Mack (1993)) to be able to conclude if using these kind of techniques in combination with micro level data can outperform the traditional method.

Even though, a single decision tree is well interpretable, using decision tree ensemble methods takes that interpretability away. Machine learning methods are well-known for their black box nature. For a non-life insurer it is valuable to know which features drive the predictions or are the most important features for the prediction of the reserves. With this information the insurer will have a better understanding in the underlying factors of the insurance risks. Therefore, we perform the interpretation method SHAP on the decision tree ensemble methods, which gives the most important features for the prediction of the reserves obtained from the decision tree ensemble methods.

For our research we use a simulated data set obtained from Gabrielli and V. Wüthrich (2018). The data set consists of 500,914 individual claims data with specific claims information and claims payments per period. We conduct three decision tree ensemble methods on the data set, namely XGBoost, Random Forest and Extra Trees to be able to answer the research question: "Can decision tree ensemble methods in combination with micro level data outperform the traditional Mack chain ladder method for the estimation of claims reserving? And if so, which method has the best results?". In our research we focus on RBNS (Reported But Not Settled) claims, explained in the next Section. The parameters of the ensemble methods are all tuned via hyper parameter tuning and the model recursively builds the full development period of the claims. For each development period, the training and test sets are split via a new kind of splitting method, similar to Baudry and Robert (2019). Furthermore, the results of the decision tree ensemble methods are evaluated with the Root Mean Squared Error (RMSE). At last, to be able to interpret the results obtained from the decision tree ensemble methods, the interpretation method SHAP is performed.

We find among the three decision tree ensemble methods, that the XGBoost algorithm outperforms the Random Forest and the Extra Trees algorithm. It also has the fastest computation time among the three methods. When comparing the predictions obtained from the XGBoost algorithm with the predictions obtained from the traditional Mack chain ladder method, we find that the predictions from the Mack chain ladder method are closer to the actual reserves than the XGBoost algorithm. However, a disadvantage of the Mack chain ladder method is that we do not know the underlying factors that play a roll in the estimation of the claims reserves. With the interpretation method SHAP we find that the age of the policyholder, the quarter of the year in which the accident took place, the number of development years between the training and test set and a specific injured body part are the most important variables for the prediction of individual claims reserving using all the three decision tree ensemble methods. This information is very valuable for non-life insurers, which they can not retrieve from the traditional triangular methods.

In the next section of this Chapter follows more background information about claims reserving. Then, in Section 1.2 previous work of past researchers on the topic will be discussed. Furthermore, in Section 2 we discuss and analyse the data set. Moreover, Section 3 elaborates on the methods used in our research, including the traditional triangular method, the decision tree ensemble methods, the construction of training and test sets and the interpretation method SHAP. Then, the results are presented in Section 4 and finally the conclusion of our research can be found in Section 5.

1.1 Background information

The following paragraph explains the building blocks of an individual claim process. In Figure 1 a timeline of a claim k is shown.

A claim starts when an accident of the policyholder takes place, this time is also known as the accident time or the occurrence time t_o as shown in Figure 1. This accident may lead to financial charges covered by a non-life insurer. In most situations, such as a car collision, the accident will not be reported at the same time as the accident takes place. The time between the occurrence time of the accident t_o and the reporting time of the claim t_r is also referred to as the reporting delay. When the reporting takes place, the insurer is able to observe details about the type of accident and the policyholder. With this information the insurer can already make a first evaluation of the total financial loss. After the accident is reported to the insurer, the claim will not be settled immediately. The time between the reporting time of the claim t_r and the settlement of the claim t_s is called the closing delay. During the closing delay, situations such as lawsuits, investigations or payments take place. The time of payments on the timeline in Figure 1 are denoted as t_{p_1}, t_{p_2}, ... , t_{p_m}, assuming that claim k needs a total of m payments. These payments will be paid until the claim is settled (or: closed) at the settlement time t_s. At this time all the information of the claim is available, including the amount of payments in each development period after the occurrence time, and the insurer is not expected to make more payments. It could also be the case that a claim re-opens after it has been settled. However, for simplicity we only consider claims that are not reopened after settlement, as shown in Figure 1.

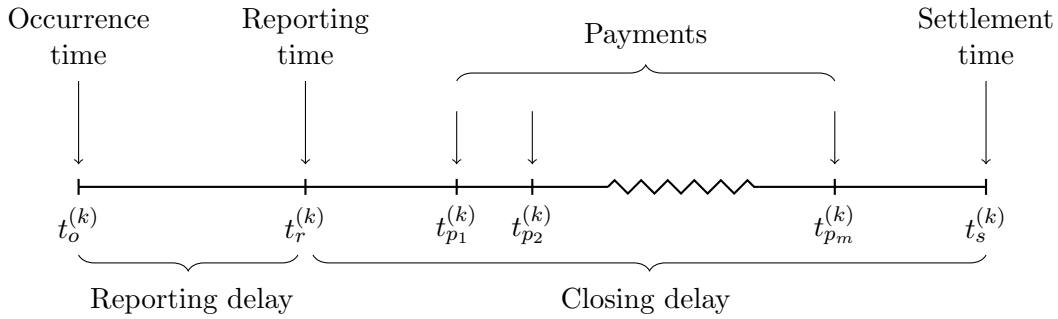


Figure 1: Timeline of a claim k.

Note: This figure shows a simplified timeline of a claim k from occurrence of the claim till settlement of the claim. The reporting delay is the time between the occurrence time and the reporting time and the closing delay is the time between the reporting time and the settlement time. In the closing delay, payments take place in several periods.

Furthermore, the reserves of an insurer can be divided into two main components as shown in Figure 2. First of all, an Incurred But Not Reported (IBNR) claim is a claim, where the accident has already occurred but the claim has not yet been reported to the insurer. A claim is an IBNR claim, when the calculation time t^* is between the occurred time t_o and the reporting time t_r, also denoted when, $t_o < t^* < t_r$. For such type of claim, information about the policyholder and type of accident are not yet available. Therefore, the insurer does not know that these IBNR claims exist. An IBNR claim is shown in the upper timeline in Figure 2.

Secondly, a Reported But Not Settled (RBNS) claim is a claim that has been reported but not yet been closed. The calculation time of the claim t^* is between the reporting date and the settlement date, also known as the period called the closing delay. Information such as the type of accident

and the policyholder are available and therefore this information can be used to estimate the reserves needed for this claim. So, if $t_r < t^* < t_s$, we are working with RBNS claims as shown in the second timeline in Figure 2.

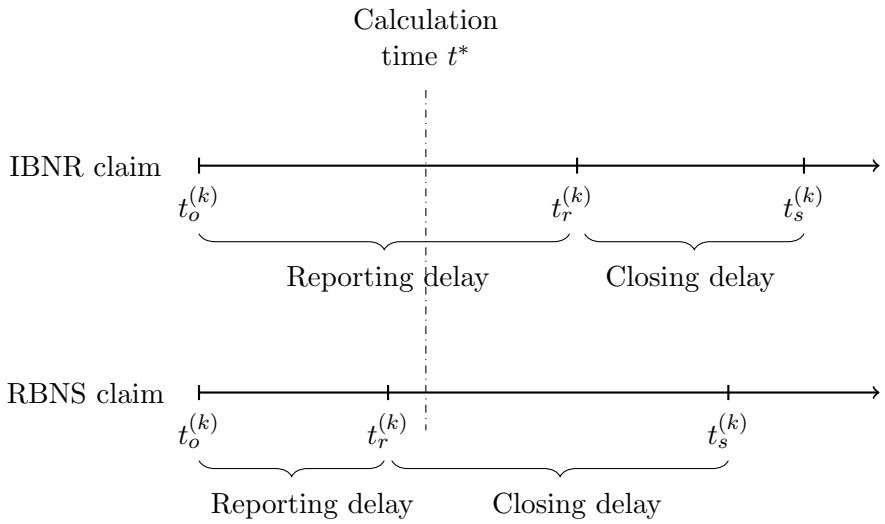


Figure 2: Distinction between an IBNR and a RBNS claim k.

Note: This figure shows an IBNR claim and a RBNS claim. IBNR claims are claims, where the calculation time takes place during the reporting delay. On the other hand, RBNS claims are claims, where the calculation time takes place during the closing delay. For IBNR claims information about the policyholder is not yet available, whereas for RBNS claims such information is available.

1.2 Previous Research

Past researchers have already published several papers surrounding the use of decision tree ensemble methods for the estimation of individual loss reserving.

First of all, Duval and Pigeon (2019) propose a Gradient Boosting-Based Approach (Chen and Guestrin, 2016) on micro-level using decision trees as weak learners. They contrast their results with the traditional aggregated techniques at macro level. They conduct five different boosting models, varying from different response variables and covariates. Their choice of a gradient boosting decision-tree model is motivated by its strong performance on structured data and its short calculation time in comparison with other methods, such as Random Forest. Through their case study, they show that using generalized linear models on micro level could be unstable for the estimation of loss reserving. However, combining an approach of a macro level model for the completion of open claims and a gradient-boosting model on micro level could be interesting for a non-life insurer. A point of future research states to compare results obtained from different machine learning methods on the same data set.

On the other hand, Wüthrich (2018a) proposes a machine learning method for individual loss reserving using regression trees (Breiman et al., 1984). For this approach, in contrary with the paper above, Wüthrich (2018a) only considers the number of payments instead of the claims amount paid. Regression trees are very flexible and allows to incorporate any kind of feature information. However, he also mentions some drawbacks, such that regression trees are not a very robust method. Using other machine learning techniques, like random forest, neural networks or a bagging approach, can overcome such issues. However, the latter can cause issues with the computation time.

Moreover, Baudry and Robert (2019) use the Extra Trees algorithm for individual claims reserving. They propose a model that recursively builds the full development of a claim, development period after development period. After that, they evaluate their performance with its true values and on the traditional chain ladder method. The paper concludes that the machine learning estimators are more robust for changes in the development patterns of claims than the chain ladder estimators based on aggregated data and that the machine learning estimates outperform the chain ladder method. Baudry and Robert (2019) states that taking into account the heterogeneity between claims causes these accurate predictions. However, they do mention that using random forests or a boosting approach, instead of the Extra Trees algorithm, could also result into great performance.

Even though all these papers use different approaches for individual claims reserving. Their similarity is that they successfully use individual claims data in the estimation process of claims reserving with a decision tree ensemble method. Also, they all state that using claims data on micro level helps to better understand the structural differences between claims payments, captures the specific development patterns of the claims and allows for claims heterogeneity. This concludes that using information on micro level is highly valuable in comparison with the traditional methods using macro level data and that machine learning techniques (especially decision tree ensemble methods) can definitely help with incorporating such data in the estimation of claims reserves.

In our research, we build three different decision tree ensemble methods using XGBoost, Random Forest and Extra Trees. In contrary to the work of Wüthrich (2018a) and Duval and Pigeon (2019), our models will recursively build the full development of RBNS claims, period after period, similar to the approach in Baudry and Robert (2019). However, Baudry and Robert (2019)'s work estimates the IBNR claims recursively via frequency and severity measures and considers a different construction of training and test sets per period. Also, we implement hyper parameter tuning to obtain the optimal parameter set and do not average out the predictions over the considered models, like Baudry and Robert (2019) do.

Furthermore, we perform the interpretation method SHAP on the predictions of the decision tree ensemble methods to make the methods interpretable. The machine learning methods used in the papers discussed above end up with a black box model, meaning that it is not known which features are the most important for the prediction of the reserves. Therefore, our research is valuable to the literature and investigates one of the future points of Duval and Pigeon (2019) to compare the results of individual claims reserving from different machine learning methods on the same data set.

2 Data

In this thesis we use the simulated data set of Gabrielli and V. Wüthrich (2018). Gabrielli and V. Wüthrich (2018) use a stochastic simulation machine based on real non-life insurance data. The simulation machine generates individual claims histories of non-life insurance claims based on neural networks for the incorporation of individual claims information, such as line of business and age of the policy holder. This information should be able to influence the reporting and settlement delay of the claim and the claim amount paid. The resulting simulated data set should be as realistic as possible, such that it reflects a real insurance claims portfolio. The true data used for the simulation is kept completely anonymous, therefore we do not know the origin of the claims. We do know that the chosen data has been preprocessed correcting for wrong values and dropping out claims with missing features. After this, the final true data set consists of almost 10 million individual claims histories on which the simulation is based on.

The simulation machine consists of eight steps in the modeling process. Gabrielli and V. Wüthrich (2018) use different simulations for the reporting delay, the payment indicator, the number of payments, the total claim size, the number of recovery payments, the recovery size, the cash flow and the claim status. Each of these steps are based on neural networks (Anthony and Bartlett, 2009). Neural networks have been proved to be very capable for regression and classification problems. However, a drawback of neural networks is the black box nature between the inputs and the output variable giving a disadvantage in the interpretation of the method. But, this drawback is not a problem for this simulation machine, because due to the missing interpretation the generating mechanism of the true data can not easily be retrieved. For a detailed description of the simulations we refer to the paper of Gabrielli and V. Wüthrich (2018).

Also, the paper states that these simulated claims perfectly allow to test the traditional reserving methods, such as the chain ladder method, as well as new developed methods on micro level data. Gabrielli and V. Wüthrich (2018) believes that with their simulation machine the drawback of no publicly available individual claims data is resolved and that it benefits the research in the field of individual claims reserving by providing a realistic synthetic data set.

The simulated data set has already been used for the estimation of individual claims reserving using neural networks by Wüthrich (2018b). Wüthrich (2018b) succeeds in using the simulated data set for the estimation of individual claims reserving using neural networks and concludes that incorporating individual claims data (in combination with neural networks) makes room for heterogeneity and captures changes in the portfolio. Since the purpose of our research is to also incorporate individual claims information using other machine learning methods, this data set is suitable for our research.

2.1 Data analytics

In this Section we elaborate more on the data analytics of the simulated data set of Gabrielli and V. Wüthrich (2018). The simulation machine simulates 500,914 claims from accident years 1994 until 2005. The data set consists of 32 variables with the following information:

- the claims number (`ClNr`), which is an explicit claims identifier related to a specific accident.
- the line of business (`LoB`), which is a categorical variable {1,...,4} indicating the line of business of the policy holder.

- the claims code (`cc`), which is a categorical variable {1,...,53} denoting the labor sector of the policy holder.
- the accident year (`AY`), which is the year the accident occurred ranging from {1994,...,2005}.
- the accident quarter (`AQ`), which is the quarter the accident of the claim took place indicated with {1,...,4}.
- the age (`age`), which is the age of the injured policyholder ranging from {15,...,70}.
- the injured body part (`inj_part`), which is a categorical variable denoting the injured body part of the policy holder with labels in {10,...,99}.
- the reporting delay (`RepDel`), which indicates in years how long the reporting delay is of the claim. The reporting delay plus the accident year is also known as the reporting year, which can be in {1994,...,2016}.
- the payments (`PayXX`), which indicates the payments made in each development period XX with XX ranging from 00 till 11.
- the open/closed indicator (`OpenXX`), which is a binary variable with 1, indicating the claim is still open in development period XX and with 0, indicating the claim is closed in development period XX with XX ranging from 00 till 11.

As explained above, the accident years range from 1994 till 2005. This means that, there are 0 till 11 development periods with 0 meaning that the payments of the claim were paid in the same year as the accident year. We assume that 11 development periods are enough to fully capture the development of the claim and therefore we consider this data set as a complete data set with no further claims to be considered surpass the 11 development periods.

As mentioned in Section 1, we use methods that recursively build the full development of the claims. This is done per calculation year, which in our case ranges from 2006 till 2016. These years are also known as development period 1 till development period 11 based on the latest accident year 2005. The calculation year is the year of which we calculate the reserves for. Depending on the accident year of the claims, the calculation year can be a different development period for each claim. For example, if we calculate the reserves for the calculation year 2006, for the claims with an accident year 2004 these are the reserves for development period 2, but for the claims with an accident year 2005 these are the reserves for development period 1.

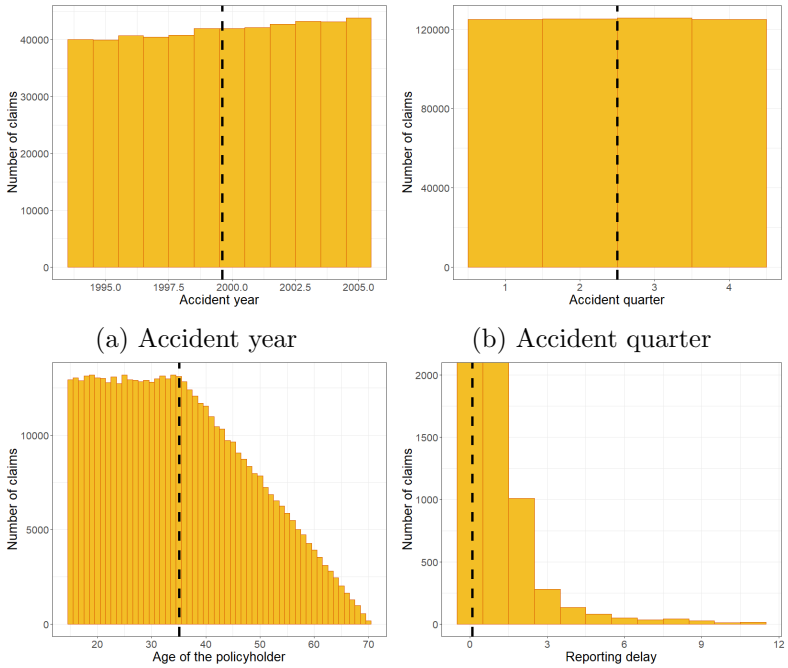
In Section 1.1, we gave more background information about the timeline of a claim and introduced the terms IBNR (Incurred But Not Reported) and RBNS (Reported But Not Settled) claims. We will explain later in Section 3.2 the construction of the training and test sets. However, in Table 1 the number of IBNR and RBNS claims per calculation year (in our specific case 2006 till 2016) for the training and test sets are already shown. Note that, in Section 1.1 we mentioned that we do not have the data for IBNR claims, since these claims are not yet reported to the insurer in the specific calendar year. The insurer does not know that these claims exist, let alone know the individual data of these claims. Therefore, using micro level data for the estimation of individual loss reserving is not allowed for the IBNR claims and it is necessary to take out these IBNR claims when conducting our analysis. For each calculation year we find a very small number of IBNR claims (less than 0.05%) as shown in Table 1. The majority of the total claims are RBNS claims and therefore it is acceptable in our research to just focus on the RBNS claims.

Table 1: Amount of IBNR and RBNS claims per calculation year used in the training and test sets.

	Test set				Training set			
Calculation year	IBNR claims	RBNS claims	Total claims	%IBNR of total claims	IBNR claims	RBNS claims	Total claims	%IBNR of total claims
2006	210	460,681	460,891	0.046%	111	456,977	457,088	0.024%
2007	97	420,863	420,960	0.023%	43	413,911	413,954	0.010%
2008	67	380,193	380,260	0.018%	19	370,685	370,704	0.005%
2009	46	339,780	339,826	0.014%	6	327,977	327,983	0.002%
2010	32	298,977	299,009	0.011%	0	285,841	285,841	0.000%
2011	24	257,020	257,044	0.009%	0	243,870	243,870	0.000%
2012	10	215,063	215,073	0.005%	0	201,905	201,905	0.000%
2013	4	172,927	172,931	0.002%	0	161,088	161,088	0.000%
2014	1	130,209	130,210	0.001%	0	120,654	120,654	0.000%
2015	0	86,960	86,960	0.000%	0	79,954	79,954	0.000%
2016	0	43,826	43,826	0.000%	0	40,023	40,023	0.000%

Note: This table shows the amount of IBNR claims, RBNS claims, total amount of claims and the percentage of the total claims that are IBNR for each calculation year of our training and test sets. The total amount of claims is the sum of the IBNR and RBNS claims and the percentage of IBNR claims is calculated as: Amount of IBNR claims / Total amount of claims $\times$ 100.

Furthermore, Figure 3 shows histograms of the number of total claims per accident year (`AY`), accident quarter (`AQ`) , age of the policyholder (`age`) and the reporting delay (`RepDel`). From Figure 3, it can be shown that for each accident year and each accident quarter roughly the same number of total claims is observed. Therefore, as expected the mean of these two variables lays perfectly in the middle of the range. Furthermore, for the age of the policyholder we find a linear decrease of the number of claims from age 35 on wards. Before age 35 the number of claims observed is roughly the same. The average age of the policyholder is 35. Next, the reporting delay of the claims finds a sharp decrease after a reporting delay of 2. For a reporting delay of 1 and 2 the number of claims exceed the limits of the Figure. The number of claims are for these two reporting delays respectively 460,340 and 38,889. The mean of the reporting delay is also very low, namely 0.09. This means that on average a claim is reported after 1 year.



(a) Accident year (b) Accident quarter

(c) Age of the policyholder (d) Reporting delay

Figure 3: The number of claims per accident year, accident quarter, age of the policyholder and reporting delay.

Note: This figure shows the number of claims per accident year, accident quarter, age of the policyholder and reporting delay. The black dotted line shows the mean of the variable. For a reporting delay of 1 and 2 the number of claims exceed the limits of the figure. The number of claims when the reporting delay is 1 and 2 is respectively 460,340 and 38,889.

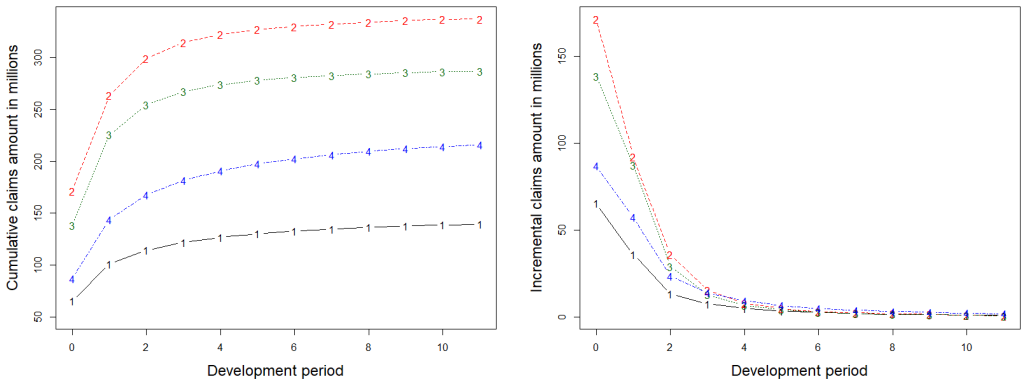
Table 2 shows the descriptive statistics of the other non-categorical and non-binary variables in the data set, namely the variables indicating the payments of the claim per development period (`PayXX`). The descriptive statistics are calculated on the total number of claims. We find a descending mean from development period 0 until development period 11. This is not a surprising result, since most claims are reported after one year or less. Thus, most claims are paid in the first few development periods. Also, for development periods 1 till 11 we find a negative claim as minimum value, meaning that the insurer gets a compensation from the policyholder.

Table 2: Desciptive statistics of the simulated data set by Gabrielli and V. Wüthrich (2018).

Variable	Mean	Min	Max	S.D.
`Pay00`	921.90	0	876,373	5387.97
`Pay01`	544.00	-10,317	1,363,113	6391.65
`Pay02`	202.70	-34,734	716,645	3634.74
`Pay03`	98.80	-82,376	343,173	2049.06
`Pay04`	57.22	-117,754	226,842	1421.74
`Pay05`	37.79	-76,607	192,709	1037.24
`Pay06`	26.41	-63,930	142,981	840.04
`Pay07`	20.12	-87,236	105,965	699.87
`Pay08`	15.84	-32,112	95,737	547.32
`Pay09`	13.52	-96,405	96,545	500.98
`Pay10`	10.40	-92,039	87,722	465.20
`Pay11`	7.10	-157,488	84,970	516.44

Note: This table shows the descriptive statistics of the payments of the claims for development period 0 till 11. The mean, minimum, maximum and standard deviation (S.D.) of the variables are calculated. The descriptive statistics are calculated on the total number of claims, so IBNR plus RBNS.

A variable indicating the line of business of the claim is also available. In Figure 4 the development of the cumulative and incremental claims amount paid are shown per line of business. The line of business is indicated with a 1, 2, 3 or 4. However, since we use a simulated data set, it is unknown what kind of line of business are referred to the categorical variable. From this Figure it can be shown that claims corresponding to the second line of business have higher to be paid claims than the other three categories. Furthermore, for all line of businesses, it can be said that the further into the development periods the lower amount of claims are paid. This can be concluded by the decreasing lines in the second figure presenting the incremental claims amounts. We also observe that the claims in each business line follow the same pattern.



(a) Cumulative claims amounts (b) Incremental claims amounts

Figure 4: Development of the cumulative and incremental claims amounts paid per line of business.

Note: This figure shows the cumulative and incremental claims amounts paid per business line in millions. Each business line is distinguished by 1,2,3 or 4. The development period goes from 0 till 11, with 0 meaning that the claims are paid in the same period as the accident year.

3 Methodology

In this Section, we discuss the methods used in this thesis. First, we explain the traditional Chain Ladder method. Afterwards, we elaborate on the construction of training and test sets to recursively build the development of the claims and the grid search for hyper parameter tuning. Then, we discuss the decision tree ensemble methods: XGBoost, Random Forest and Extra Trees, which are used to estimate the individual claims reserving. At last, we elaborate on the interpretation method SHAP.

3.1 Mack Chain Ladder

In Section 1 we mentioned that the traditional triangular method use aggregated data for the estimation of claims reserves. One of the first models based on these triangular methods is proposed by Mack (1993), also called the Mack Chain Ladder method. Due to its simplicity and its distribution-free approach, the traditional triangular method is still widely used by non-life insurers for the estimation of their claims reserves. However, this method does not make use of individual claims data and therefore solely looks at aggregated data. We use Mack Chain Ladder method as a benchmark method for comparison with the machine learning techniques on micro level proposed later in this Section.

First, lets introduce C_{ij} to denote the accumulated claims amount paid of accident year i up

until development period j, where $i = 1, ..., I$ and $j = 0, ..., J$. The number of accident years and development periods is equal to each other, thus we can denote J as $I - 1$. For example, if the first accident year in the data set is 1994 and we are calculating the accumulated claims amount paid for development period 0, this means that we are summing up the total claims amount paid in 1994, denoted as C_{10}. Furthermore, for development period 1 (and the same accident year 1994), the accumulated total claims amount paid up until development period 1, is denoted as C_{11} and are the accumulated total claims amount paid up until 1995 (so also including 1994) for all the claims with an accident year of 1994.

We can split the accumulated total claims amount paid C_{ij} into two different parts, namely the observed claims amounts and the predicted claims amounts. In Table 3 a basic chain ladder method is shown with on the rows the accident years and on the columns the development periods. The diagonal in this Table refers to a calendar year. When $j \leq J - i + 1$, we can observe the accumulated total claims amount paid for each i and j. In Table 3, these claims are in the upper left triangle part of the table. The goal is to ultimately estimate the outstanding claims reserves as:

$$\hat{R}_i = \hat{C}_{i,J} - C_{i,J+1-i}, \tag{1}$$

for each accident year $i = 2, ..., I.$, where $\hat{C}_{i,J}$ are the predicted accumulated total claims amount paid as shown as the last column in the lower triangle part of Table 3 and $C_{i,J+1-i}$ the observed accumulated total claim payments, also referred to the diagonal value of accident year i.

Table 3: A basic chain ladder method.

Accident year i	Development period j							
	0	1	2	..	j	..	..	J
1								
2				Observed				
3				C_{ij}				
..								
i								
..						Predicted		
..						$\hat{C}_{ij}$		
I								

Note: This table shows a representation of a basic chain ladder method. The upper left triangle of the table shows the observed accumulated total claim payments and the lower right triangle are the predicted accumulated total claim payments. The diagonal refers to a calendar year.

In the basic chain ladder method, Mack (1993) assumes that there are development factors $f_j > 0$ for each $j = 0, ..., J - 1$ with

$$E(\hat{C}_{i,j+1}|C_{i0}, C_{i1}, ..., C_{iJ}) = f_j C_{ij} \quad \text{or} \quad \hat{C}_{i,j+1} \approx f_j C_{ij}, \tag{2}$$

for every accident year $i = 1, ..., I$. According to the chain ladder method, the accumulated total claim payments over the development periods behave in a similar pattern. This pattern should be captured by the development factors, also referred to as age-to-age factors or the chain ladder factors.

Ultimately, we need to estimate $\hat{C}_{i,J}$ for each $i = 2, ..., I$ to be able to calculate the reserves as in

Equation 1. This can be calculated as

$$\hat{C}_{i,J} = C_{i,I-i} \prod_{j=I-i}^{J-1} f_j \quad \text{for all } i = 2, ..., I. \tag{3}$$

Furthermore, the development factors f_j are calculated as described in Mack (1993) with

$$f_j = \frac{\sum_{i=1}^{I-j-1} C_{i,j+1}}{\sum_{i=1}^{I-j-1} C_{i,j}} \quad \text{for all } j = 0, ..., J-1. \tag{4}$$

Additionally, with this approach of estimation it can also be assumed that the accident years are independent from each other.

3.2 Construction of training and test sets

In this Section we elaborate on the construction of training and test sets, which are necessary to recursively build the development of the claims with the decision tree ensemble methods discussed later in this Chapter.

First, we re-arrange Table 3 with the development years on the columns. Note that the development years are not equal to the development periods j as presented in Table 3. The development year is the calendar year in which the claim has been paid. Therefore, for each accident year not all development years are applicable and there are different development years for each accident year as shown in Table 4. However, the number of development periods remains the same for each accident year. In Table 4 an example is shown with $i = 1, .., 6$ accident years and $j = 0, .., 5$ development periods for each accident year. The light grey area in Table 4 are the observed accumulated claim payments C_{ij} and the dark grey area are the predicted accumulated claim payments $\hat{C}_{ij}$, predicted with the decision tree ensemble methods later explained in this Chapter. Note that the example given in Table 4 only consists of six accident years and five development periods. In our data set we have data available from twelve accident years ranging from 1994 till 2005 as explained in Section 2 and we work with eleven development periods.

Table 4: Representation of the accumulated claim amount per development year.

	Development year										
Accident year i	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010
	Observed C_{ij}						Predicted $\hat{C}_{ij}$				
2000 ($i = 1$)	C_{10}	C_{11}	..	..	..	C_{15}	×	×	×	×	×
2001 ($i = 2$)	×	C_{20}	C_{21}	..	..	..	$\hat{C}_{25}$	×	×	×	×
2002 ($i = 3$)	×	×	C_{30}	C_{31}	..	..	..	$\hat{C}_{35}$	×	×	×
2003 ($i = 4$)	×	×	×	C_{40}	C_{41}	..	..	..	$\hat{C}_{45}$	×	×
2004 ($i = 5$)	×	×	×	×	C_{50}	C_{51}	..	..	..	$\hat{C}_{55}$	×
2005 ($i = 6$)	×	×	×	×	×	C_{60}	$\hat{C}_{61}$	..	..	..	$\hat{C}_{65}$

Note: This table shows a representation of the accumulated claim number per development year for each accident year $i = 1, .., 6$. The light grey area represents the observed accumulated claim payments and the dark grey area represents the predicted accumulated claim payments. The cells with a cross can be ignored, since these are not observed nor predicted.

For example, if the calculation year $t^* = 2006$, this means that for accident year 2001 ($i = 2$) we predict the claim payments for development period 5 ($\hat{C}_{25}$) and for accident year 2002 ($i = 3$) this is

for development period 4 ($\hat{C}_{34}$) etc. The goal is to predict the claim payments for each development year 2006 until 2010, these are $J = 5$ development years after the last accident year 2005. For each of these development years, we need a separate training and test set. As a result of that, we also need a separate model for each of these calculation times.

We continue with the example with the calculation year $t^* = 2006$, in Table 5 this is denoted as the dark blue coloured column. To be able to predict the accumulated total claim payments for this calculation year, we construct explicit training and test sets. We refer to the light orange area as the X.TRAIN area and the dark orange area as the Y.TRAIN area. These two training sets are used for the training of the model with a decision tree ensemble method explained later in this Chapter. For the testing of this trained model, we then use the light blue area also referred to as X.TEST. This model is then used for the prediction of the accumulated total claim payments for development year 2006, in this case known as Y.TEST. The training sets (X.TRAIN and Y.TRAIN) use all the claim data from 2000 until 2004 and the test sets use (X.TEST and Y.TEST) all the claim data from 2001 till 2005 (in this example).

However, in Section 1.1 we mentioned that information about IBNR claims are not yet available at the calculation time. For example, if a claim is occurred in 2004, but reported in 2007, we can not use the data of this claim for the training and test sets for the calculation year 2006, because this data is not available in 2006. Therefore, we omit all the IBNR claims in the construction of the training and test sets and solely focus on RBNS claims with the use of decision tree ensemble methods.

Table 5: Construction of training and test sets for $t^* = 2006$.

	Development year										
Accident year i	2000	2001	2002	2003	2004	2005	**2006**	2007	2008	2009	2010
	Observed C_{ij}						Predicted $\hat{C}_{ij}$				
2000 ($i = 1$)	C_{10}	C_{11}	..	..	..	C_{15}	×	×	×	×	×
2001 ($i = 2$)	×	C_{20}	C_{21}	..	..	..	$\hat{C}_{25}$	×	×	×	×
2002 ($i = 3$)	×	×	C_{30}	C_{31}	..	..	..	$\hat{C}_{35}$	×	×	×
2003 ($i = 4$)	×	×	×	C_{40}	C_{41}	..	..	..	$\hat{C}_{45}$	×	×
2004 ($i = 5$)	×	×	×	×	C_{50}	C_{51}	..	..	..	$\hat{C}_{55}$	×
2005 ($i = 6$)	×	×	×	×	×	C_{60}	$\hat{C}_{61}$	..	..	..	$\hat{C}_{65}$

	Development year										
Accident year i	2000	2001	2002	2003	2004	2005	**2006**	2007	2008	2009	2010
	Observed C_{ij}						Predicted $\hat{C}_{ij}$				
2000 ($i = 1$)	C_{10}	C_{11}	..	..	..	C_{15}	×	×	×	×	×
2001 ($i = 2$)	×	C_{20}	C_{21}	..	..	..	$\hat{C}_{25}$	×	×	×	×
2002 ($i = 3$)	×	×	C_{30}	C_{31}	..	..	..	$\hat{C}_{35}$	×	×	×
2003 ($i = 4$)	×	×	×	C_{40}	C_{41}	..	..	..	$\hat{C}_{45}$	×	×
2004 ($i = 5$)	×	×	×	×	C_{50}	C_{51}	..	..	..	$\hat{C}_{55}$	×
2005 ($i = 6$)	×	×	×	×	×	C_{60}	$\hat{C}_{61}$	..	..	..	$\hat{C}_{65}$

Note: This table shows the construction of the training and test sets for the calculation time $t^* = 2006$. The yellow area (X.TRAIN) denotes the data we use for the training of the model. The orange area (Y.TRAIN) is the dependent variable we use for the training of the model. The light blue is (X.TEST) the data used in the trained model for predicting the accumulated total claim payments in 2006, denoted as the dark blue area Y.TEST).

Furthermore, when the calculation year t^* shifts to 2007 the training and test sets also shift. Y.TEST becomes the predictions in the column of 2007, X.TEST consists of all the data with an

accident year of 2002 and higher, Y.TRAIN contains the accumulated total claim payments up until 2005 for the claims with accident years 2000 until 2003 and X.TRAIN consists of all the data with an accident year from 2000 until 2003. In the Appendix in Table 10, you can find an example of the construction of training and test sets for the calculation year 2007. All the other training and test sets for the other calculation times are constructed in the same way.

As explained in Section 2, we use the simulated data set from the stochastic simulation machine of Gabrielli and V. Wüthrich (2018). For the X .TRAIN and X.TEST sets, we use the independent variables: line of business (`LoB`), claims code (`cc`), accident quarter (`AQ`), age (`age`), injured body part (`inj_part`) and the number of development years between the Y.TRAIN and Y.TEST set respectively, and the accident year. For the X.TRAIN set, the latter independent variable is 2005 minus the corresponding accident year for any calculation time in the example of Table 5, since Y.TRAIN (the orange column) is always corresponding to the accumulated total claim payments of 2005. In contrary to the Y.TRAIN set, the calculation time of the Y.TEST set (dark blue column) does change per construction of the training and test sets. However, the number of development years between the calculation time of the corresponding Y set and the accident year, is never the same for the training and test sets for the same calculation year. Thus, we use distinct data in both training and test sets and do not re-use data for both training and testing. For the categorical variables line of business (`LoB`), claims code (`cc`) and injured body part (`inj_part`), we use dummies for each value in these variables. This means that both the X.TRAIN and X.TEST set consist of 104 variables. Note that not all values in the range explained in Section 2 are used for the variables. For example, for the injured body part, the values 82 until 98 are not used. Therefore, we also do not use a dummy variable for these values, because the columns will only consist of zeros.

3.3 Hyper parameter tuning and approach for training and testing

In this Section we explain the use of hyper parameter tuning for our decision tree ensemble methods that are discussed in the next Section and our approach for training and testing the methods to be able to estimate the full development of the claims.

After constructing the training and test sets for each calculation year as explained in Section 3.2, we perform hyper parameter tuning on the parameter set of the decision tree ensemble methods. For the hyper parameter tuning we first set a grid with the parameters and its values to be optimized during the tuning process. After this we run the model on each combination of parameter values in the grid and evaluate the models with the Root Mean Squared Error (RMSE), calculated as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} |(\hat{y}_{ij} - y_{ij})|^2}{n}}, \tag{5}$$

where $\hat{y}_{ij}$ stands for the predicted reserves and y_{ij} for the actual reserves for development period j. The model with the lowest RMSE is the optimal model and therefore the parameter set corresponding to this model is the optimal hyper parameter set. This process is further elaborated in Algorithm 1.

Each decision tree ensemble method has different parameters to be optimized, so each grid is set on different values. The hyper parameter tuning is an important step in our approach. Some important reasons for using hyper parameter tuning to machine learning methods are that it reduces human effort, since developers spend a considerable amount of time finding the optimal parameters, it has a

significant impact on the performance of the model and it makes the models better reproducible for other researchers (Schratz et al. (2019), Yang and Shami (2020)).

As explained in Section 3.2, for each of the calculation years (the development years that need a prediction for their claim payments) we have separate training and test sets, separate models and therefore also separate optimal hyper parameter sets for each of these models. In Algorithm 1, the pseudo code for the estimation of individual claims reserving with a machine learning method is shown. This pseudo code shows the use of the different training and test sets for each calculation year. The different training and test sets are used to find the optimal hyper parameter set in the grid search. Moreover, the pseudo code also shows the testing of the final models to estimate the full development of the claims.

Algorithm 1 Pseudocode for estimating the full development of the individual claims reserving

Input: `J` (number of development periods), `list_X_train`, `list_Y_train`, `list_X_test` (lists of all the training and test sets for the J development periods) and a machine learning method

Output: The full development of all the RBNS claims as in Table 4

```
1:  grid ← Specify the grid with the parameters to be optimized during the grid search
2:  for j ← 1 to J do
3:      all_rmse ← Create an empty vector (the RMSE will be saved in this vector)
4:      for g ← 1 to length(grid) do
5:          ML_model ← Train the model with a machine learning method using list_X_train[j],
6:                     list_Y_train[j] and the parameters in grid[g]
7:          Y_pred ← Predict the claims for j with ML_model and list_X_test[j]
8:          acc_Y_pred ← Accumulate Y_pred per accident year
9:          res_pred ← Calculate the reserves using acc_Y_pred, the observed accumulated claim
10:                    payments and if j > 1 the predictions of the prior calculation years
11:         rmse ← Calculate Root Mean Squared Error between res_pred and the actual reserves and
12:                save this result in the vector all_rmse
13:     end for
14:     opt_param ← Get the optimal hyper parameter set, which is the index in grid
15:                 with min(all_rmse)
16:     ML_model_opt ← Train the model with a machine learning method using list_X_train[j],
17:                    list_Y_train[j] and the optimal hyper parameter set opt_param
18:     Y_pred_opt ← Predict the claims for j with ML_model_opt and list_X_test[j]
19:     acc_Y_pred ← Accumulate Y_pred_opt per accident year
20:     res_pred ← Calculate the reserves using acc_Y_pred, the observed accumulated claim
21:                payments and if j > 1 the predictions of the prior calculation years
22: end for
```

Note: This algorithm contains the pseudo code for the estimation of the full development of the individual claims reserving. As input it uses the number of development periods, the training and test sets as explained in Section 3.2 and a machine learning method.

The pseudo code starts with the input and output of the algorithm. As input for the algorithm we use the number of development periods `J`, the lists of the X.TRAIN, X.TEST and the Y.TRAIN sets, respectively noted in the algorithm as `list_X_train`, `list_X_test` and `list_Y_train`, and a machine learning method. For the machine learning methods we use decision tree ensemble methods that will be discussed in Section 3.4. However, other machine learning methods, such as Neural Networks, are also acceptable to use in this algorithm. Then, using all these variables as input for the algorithm, gives the full development of the RBNS claims as desired in Table 4.

Furthermore, line 1 shows the specification of the grid with the parameters of the machine learning method to be optimized during the grid search. Then, in line 2-4, we loop over the development periods

and the grid search specified in the previous step, such that for each development period we use the corresponding training and test for the hyper parameter tuning. We also create an empty vector at the beginning of each loop over a development period, which is necessary for later in this Algorithm. The search for the optimal hyper parameter set is shown in lines 5-12, where each of the hyper parameter sets in `grid` is used for the training of the model with a machine learning method. The output of the machine learning method is accumulated and the reserves are calculated as in Equation 1. The next step is then to calculate the Root Mean Squared Error (RMSE) as in Equation 5. Furthermore, on line 14-21 we get the optimal hyper parameter set from the `grid`, which is the hyper parameter set that gives the lowest RMSE of the trained model. We again train the model, but this time with the optimal hyper parameter set, accumulate the predictions per accident year and calculate the reserves again as in Equation 1. Repeating this process for all the development periods gives the desired output, namely the full development of all the RBNS claims.

3.4 Decision tree ensemble methods

In Section 1.2, we already delve into papers from past researchers with regards to using decision tree ensemble methods for individual claims reserving. From these papers we conclude that decision tree ensemble methods are acceptable methods for incorporating micro-level data to the estimation of claims reserving. In this Section we explain the decision tree and elaborate on the different decision tree ensemble methods used in this thesis for the estimation of individual claims reserving and for the comparison with the benchmark model as explained in Section 3.1.

Ensemble methods try to improve the results obtained by machine learning methods by combining several machine learning methods (the learning algorithms of the ensemble method) together. This allows for better performance in prediction compared to a single model (stacking), it decreases the variance (bagging) and it decreases bias (boosting). The advantage of using ensemble methods is that if one of the learners fails, the overall system can still recover the error (Valentini and Masulli, 2002). However, a disadvantage is that ensemble methods are less interpretable than a single learning algorithm and can result in a black box model.

In this research, we use decision tree ensemble methods, which are ensemble methods that use decision trees as their learning algorithms. Therefore, to be able to fully understand these ensemble methods it is necessary to first understand a single decision tree.

A decision tree can have two different types, namely classification trees for classification purposes and regression trees for regression purposes (Breiman et al., 1984). In our case, we work with a continuous decision variable (the payments of a claim) and therefore use regression trees. The idea of a regression tree is to partition the feature space by splitting the set of variables with an impurity measure. For regression trees the impurity measure is the sum of squared residuals (Loh, 2014). The split which reduces the impurity measure the most, is chosen for the next split. Each split is chosen such that it maximizes the performance of the model. The first split of the tree happens in the so-called root node, which is also known for the most important variable for the estimation of the output. An example of a decision tree can be found in the Figure below.

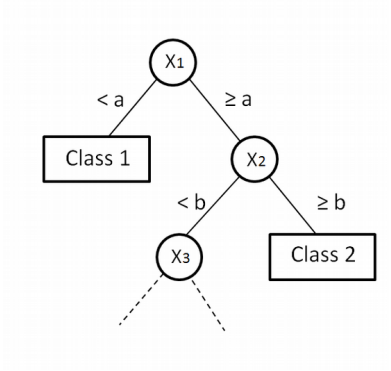


Figure 5: An example of a decision tree.
Source by: https://www.researchgate.net/figure/Decision-tree-example_fig1_320970818

Furthermore, the advantages of using decision trees is that decision trees are non-parametric, which means they do not make any assumptions about the distribution of the data. Single decision trees are also interpretable. There are several functions that can give you the most important features for the estimation of the decision variable. Some disadvantages are that decision trees are very prone to overfitting. One of the reasons we use ensemble methods is that ensemble methods can prevent this problem. For classification purposes with an imbalanced data set, decision trees can have issues with the low representation of a class. Observations of the minority class can get lost in the majority class nodes, which causes the model to give non accurate predictions of the minority class. A solution for this issue is to make the imbalanced data set more balanced with up and downward sampling (Kotsiantis et al., 2005).

3.4.1 XGBoost algorithm

The first decision tree ensemble method we use is a gradient boosting algorithm (Friedman, 2001), namely the XGBoost algorithm, also known as the Extreme Gradient Boosting algorithm by Chen and Guestrin (2016). The XGBoost algorithm has been widely used by several researchers in different areas. Schratz et al. (2019) uses the algorithm for the diagnosis of chronic kidney disease. On the other hand, Pan (2018) shows that the algorithm can also be used for predicting the concentration of air quality in China. As explained in Section 1.2, the XGBoost algorithm can also be used for the estimation of individual claims reserving (Duval and Pigeon (2019), Pesantez-Narvaez et al. (2019)). However, both of these paper do not consider the splitting of the training and test sets, such as explained in Section 3.2 and use different type of features.

The XGBoost algorithm is a scalable tree boosting system. The main idea of this algorithm is to build D decision trees (Breiman et al., 1984) after each other, such that each subsequent tree is trained using the residuals of the previous tree. In other words, each subsequent tree corrects the errors made by the previous trained tree and predicts the outcome. This process is called boosting. The algorithm is also known as an ensemble learner, meaning that it creates a final model based of a set of individual models, also known as the decision trees. The individual models have a weak predictive power and are prone to overfitting when considered separately, but combining all these models in an ensemble learner can lead to improvements in the results. The individual decision trees are not completely built on random subsets of the data, but the algorithm sequentially puts more weight on data points with high errors. In Figure 6 you can find an example of a boosting algorithm with decision trees as learners.

The algorithm makes use of the gradient descent algorithm to minimize the loss when adding a new tree, hence the name gradient boosting.

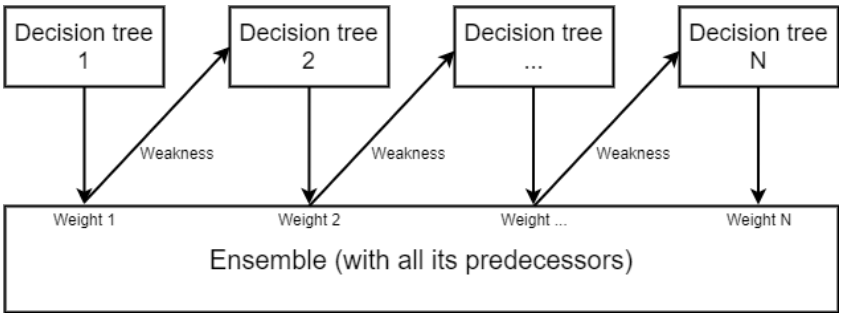


Figure 6: Boosting algorithm.

Furthermore, the XGBoost algorithm is a specific implementation of the gradient boosting algorithm (Friedman, 2001). In the Appendix you can find the gradient boosting algorithm in Algorithm 2. The differences between these two algorithms is that the XGBoost algorithm computes second order partial derivatives of the loss function, which provides more specific information about the direction of the gradient for the minimization of the loss function. Also, the XGBoost algorithm uses a more advanced regularization (L1&L2) (Ng, 2004), which improves the generalization capabilities of the model. Moreover, it can use parallel computing. All these implications cause high predictive performance and fast computation time compared to the gradient boosting algorithm. Due to this, the XGBoost algorithm dominates in many competitions, such as in the Kaggle competitions (Bentéjac et al., 2019). Therefore, we choose to use the XGBoost algorithm over a normal gradient boosting algorithm in our research.

We implement XGBoost in **R** with the package **xgboost** and perform hyper parameter tuning as explained in Section 3.3. One of the parameters that we tune is the eta parameter, which controls the learning rate at which our model detects patterns in the data (`eta`). Typically a lower eta means a larger computation time. Furthermore, the gamma parameter controlling regularization to prevent overfitting is also tuned, with a higher value meaning higher regularization and 0 indicating that no regularization is used in the algorithm. Moreover, the maximum depth of the tree (`max_depth`) is also tuned. Setting a cap to the maximum depth of the tree prevents overfitting and reduces the variance. All other parameters are set on fixed values, such as the booster is set on `gbtree`, which builds decision trees and optimizes this using regularization and gradient descent, and the number of iterations (`nrounds`) is set on 500. All other variables are set on its default values.

3.4.2 Random Forest

The next decision tree ensemble method we use is Random Forest, just like the XGBoost algorithm, this is a well known ensemble method of decision trees in the field of machine learning introduced by Breiman (2001). Just as the XGBoost algorithm as explained in the previous Section, Random Forest is also widely used in the literature. For example, for breast cancer detection among women (Elgedawy, 2017) and credit scoring (Zhang et al., 2018). Furthermore, for individual claims reserving Baudry and Robert (2019) confirm that a Random Forest instead of their implemented Extra Trees algorithm, can also be considered for prediction of individual claims reserving. Also, Wüthrich (2018a) mentions

in his paper that using Random Forest can overcome issues such as lack of robustness experienced with just regression trees.

Random Forest combines multiple decision trees with bagging and bootstrapping methods. Bagging means that the decision trees are independently grown from each other. It can be used for regression and classification purposes. For regression, we fit B regression trees to bootstrap-sampled versions of the data (Hastie et al., 2009), which are drawn from the full data set with replacement. It can be the case that these bootstrap-sampled versions of the data only contain a subset of the observations. Each $b \in B$ generates a prediction $\hat{y}_b$. The Random Forest prediction is then the average of all these predictions $\hat{y} = B^{-1} \sum_{b=1}^{B} \hat{y}_b$. In Figure 7, an example of a Random Forest prediction is shown for $B = 600$ trees. Breiman (2001) states that generating different trees and averaging their predictions is more preferred than generating a single tree, because this reduces the variance of the model and it is very difficult to find one highly-optimized decision tree. In contrary to XGBoost, Random Forest suffers large computation times and complexity in the interpretability. In the Appendix you can find the algorithm for Random Forest in Algorithm 3.

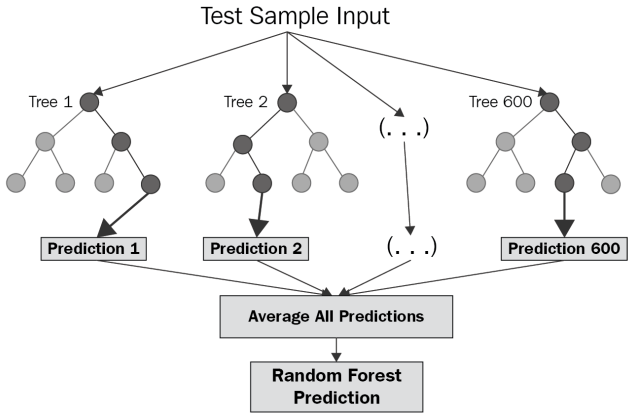


Figure 7: Random Forest algorithm.
Source by: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781789132212/2/ch02lvl1sec21/decision-tree-based-ensemble-methods

We implement Random Forest in **R** with the package **randomForest**. As explained in Section 3.3, we perform hyper parameter tuning to optimize the different parameters used in Random Forest. Due to the high computation time of Random Forest, we only consider a few parameters to be tuned, which are in our opinion the most important parameters for the model. The parameters that we optimize are the number of variables used as candidates at each split (`mtry`) and the minimum node size (`nodesize`). The number of variables used as candidates at each split is of importance, because the tree is then able to identify the effects that different variables have on the dependent variable. This prevents the tree from always being dominated by the same set of independent variables. Furthermore, the minimum node size is of importance for the depth of the tree. Setting a cap to the maximum depth of the tree prevents overfitting and reduces the variance. The larger the minimum node size, the smaller the trees are grown and the less time it takes for the algorithm. Another important variable is the number of trees (`ntree`). The number of trees contribute to the reduction of bias and variance, because of the averaging of the predictions of the trees. However, due to the high computation time of the tuning process we set the number of trees on a fixed value of 50. At last, all other variables are set on default. This means that the data is sampled with replacement, the trees are grown to the

maximum possible terminal nodes and the importance of the predictors is not assessed.

3.4.3 Extra Trees

The last tree-based ensemble method is the Extra Trees algorithm, also known as Extreme Randomized Trees algorithm developed by Geurts et al. (2006). Extra Trees have also been used by past researchers in several areas. Sharaff and Gupta (2019) use Extra Trees for the detection of spam mails and conclude that the accuracy using the Extra Trees classifier has a major improvement compared with previous researches using decision trees. Also, in the field of health care Extra Trees has been used for the prediction of cardiovascular diseases by Shafique et al. (2019). In the field of non-life insurance Baudry and Robert (2019) succeeds to use Extra Trees for the prediction of individual claims reserving. However, they use a different approach with a different construction of the training and test sets.

Just as Random Forest and XGBoost explained in the last two previous sections, Extra Trees is also suitable for both classification and regression problems. While XGBoost is a boosting method and Random Forest combines bagging and bootstrapping methods, Extra Trees is an alternative bagging process, which uses a classical top-down procedure using unpruned decision trees. Extra Trees is very similar to Random Forest and the visual representation in Figure 7 is also suitable for Extra Trees. However, the two main differences in comparison with Random Forest is that Extra Trees splits the nodes of the tree randomly and uses the whole data sample to grow trees (rather than bootstrapped-samples as used in Random Forest). The predictions of M trees are aggregated to yield the final prediction with use of arithmetic average in regression problems. Geurts et al. (2006) state that using randomization for cutting points combined with averaging out the predictions over M trees leads to reduction of the variance and using the whole data sample instead of bootstrapped-samples leads to minimization of the bias. Even though, given the tight similarities between Extra Trees and Random Forest, Ampomah et al. (2020) shows that from six different decision tree ensemble methods, the Extra Trees algorithm outperforms all other methods (also Random Forest) in the context of stock price predictions.

We make use of **R** and the package **ranger** for the implementation of the Extra Trees algorithm. Similar to XGBoost and Random Forest, we also conduct hyper parameter tuning as explained in Section 3.2. We choose to tune a few parameters taking into consideration the high computation time of the grid search. First of all, we tune the number of features tried at each node (`mtry`). This is of importance for the tree to identify the different effects the variables have on the dependent variable. If you would always choose the same variables, some variables might get dominated by other variables and this can cause issues in the estimation process. Furthermore, the size of the leaves of the tree (`nodesize`) is also tuned, this variable corresponds to the depth of the tree. The larger the size of the nodes, the smaller the trees and the faster the computation time. At last, we tune the number of random cuts for each feature (`numRandomCuts`). The higher the number of cuts, the higher the chance of a good cut. The first three (to be tuned) parameters are also tuned for Random Forest. The number of random cuts (`numRandomCuts`) is not tuned in Random Forest, since this parameter is a specific parameter only used for the Extra Trees algorithm. Just as the Random Forest, we do not tune the number of trees (`ntree`) due to the high computation time of the tuning process. However, the number of trees is still an important variable, which a higher number of trees contributes to the reduction of bias and variance. We set the number of trees on a fixed value of 50 for the Extra Trees

algorithm.

3.5 Interpretation of the methods

Machine learning methods are known for their black box nature. Based on the output of the machine learning methods, it is not immediately clear what specifically happens and how the predictions are made in such methods. Even though a method performs well, not knowing which variables drive the predictions can make the predictions untrustworthy. By explaining the predictions with the independent variables used in the methods, we provide a better understanding of the relationship between our independent variables and our predictions of the claims reserves. With the results of interpretation methods, insurers will have a better understanding in what the driving forces are in the estimation of the claims reserves.

In our case, using decision tree ensemble methods for the estimation of individual claims reserving, also leads us to interpretation problems. If we would have used a single decision tree for our predictions, the interpretation would be fairly straight forward. The variables that are split in the first few nodes of the decision tree are then the most important variables for the prediction of the model. However, using decision tree ensemble methods make the interpretation a lot harder, because you have multiple decision trees with most likely different splits in the first few nodes. Therefore, to interpret decision tree ensemble methods we need an interpretation approach that suits these type of methods the best.

Recently, researchers have proposed new approaches to solve these interpretation problems for machine learning methods. Ribeiro et al. (2016) proposed the Local Interpretable Model-agnostic Explanations (LIME) method to analyse the relationship between the independent variables and the target variable on a specific observation. For this analysis, LIME considers the local neighborhood along the particular observation being explained. Another approach in the literature is proposed by Lundberg and Lee (2017). They present a unified framework for interpreting predictions coming from complex models, such as ensemble or deep learning models, called SHAP (SHapley Additive exPlanations). SHAP uses Shapley values (Shapley, 1953) from game theory to calculate the added value of each feature to the predictions of the model. In contrary to LIME, SHAP does not only perform locally, meaning on a single observation. This means that when using LIME the interpretation results differ among every observation. In our case this is not a desirable way to interpret the results, because we work with a large portfolio of claims. Therefore, to get an interpretation for a big part of our portfolio of claims we use SHAP instead of LIME. In the next paragraph we further explain the use of SHAP.

3.5.1 Shapley Additive Explanations (SHAP)

Lundberg and Lee (2017) propose a unified feature importance measure called the SHAP (SHapley Additive exPlanations) values. This framework is a model agnostic method for explaining a model, meaning that the SHAP values can be used for many different models. The output of this interpretation method provides an understanding in the relationship between the variables used in the decision tree ensemble methods and the prediction of the claims reserves. It is not a measure for causality, but solely focuses on the intepretability of the model. Various researchers, such as Chen et al. (2018) and Lubo-Robles et al. (2020), have already used this method for the interpretation of machine learning methods. SHAP is mostly used to explain individual predictions, just as the interpretation method

LIME explained in the previous Section. However, SHAP also has the possibility to interpret the predictions globally, based on the mean of the SHAP values. Therefore, we prefer the use of SHAP over the method LIME. We use the global interpretation method of SHAP to explain a big part of our portfolio of claims.

Furthermore, SHAP is the only interpretation method that satisfies the three properties: local accuracy, missingness and consistency (Lundberg and Lee, 2017). Local accuracy means that the SHAP values of an observation must correspond to the output of the model. Therefore, the output of the interpretation method is reliable and truthful. Furthermore, missingness means if an observation has a missing value for a specific feature x, that the missing feature x does not contribute to the interpretation method. At last, consistency states that changes in the model, which then causes contributions of a specific feature to increase or stay the same, should not contribute to decreasements in the SHAP value. For a more detailed description of these three properties, we refer to the paper of Lundberg and Lee (2017).

The purpose of the SHAP framework is to explain the prediction of a single observation i by computing the added value of each feature to the prediction. As mentioned in the previous Section, SHAP uses Shapley values (Shapley, 1953) from game theory to calculate this added value. The added value of a feature x is calculated as the weighted decrease or increase in the prediction when adding the feature x over all subsets of features that excludes the feature x, denoted as $S \subseteq F \setminus \{x\}$.

To compute the added value per feature x, a model $f_{S\cup\{x\}}$ is trained with feature x in its feature space. On the other hand, another model f_S is also trained excluding feature x in its feature space. The predictions of these two models are compared with its current input values $f_{S\cup\{x\}}(z_{S\cup\{x\}}) - f_S(z_S)$, where z_S corresponds to the values of the input variables in set S. These differences are computed for all possible subsets $S \subseteq F \setminus \{x\}$. The SHAP values are then computed as the weighted average of all differences:

$$\phi_x = \sum_{S\subseteq F\setminus\{x\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!}[f_{S\cup\{x\}}(z_{S\cup\{x\}}) - f_S(z_S)]. \tag{6}$$

When interpreting more than one observation, the SHAP values are computed for every observation. The global SHAP value is then calculated as the mean of the SHAP values computed from every observation.

A drawback of using SHAP values is the extensive computation time due to the consideration of all possible subsets $S \subseteq F \setminus \{x\}$. However, researchers have found a solution to this drawback by developing model specific algorithms. These optimizations take advantage of the model's structure and significantly reduces the computation time of the SHAP values. For example, in the case of neural networks the method Deep SHAP can be used, which considers the structure of a neural network. In our case, using decision tree ensemble methods, we can improve the computation time of the SHAP values with considering the structure of a decision tree. This approach is called Tree SHAP, which is further elaborated in the next Section.

3.5.2 Tree SHAP

Lundberg et al. (2019) propose another version of the SHAP method, specifically for decision tree ensemble methods, called Tree SHAP. It has been proven that computing SHAP values for decision

tree ensemble methods without the Tree SHAP method causes inconsistency. This contradicts the three properties mentioned in Section 3.5.1 and does not contribute to a useful interpretation method.

First of all, Tree SHAP is an algorithm that calculates the SHAP values in polynomial time instead of exponential time. The intuition behind this application is to recursively monitor the proportion of the possible subsets that flow down into the leaves of a decision tree. This improves the computation time significantly and takes into consideration the structure of a decision tree.

Secondly, Tree SHAP separates interaction effects from the main effects. This separation uncovers important interactions captured by the decision tree ensemble methods that otherwise will be missed. The Shapley interaction index computed in the Tree SHAP method is then calculated as:

$$\Phi_{\{x,y\}} = \sum_{S \subseteq F \setminus \{x,y\}} \frac{|S|!(|F| - |S| - 1)!}{2(|F| - 1)!} \nabla_{\{x,y\}}(S), \tag{7}$$

where $x \neq y$ and

$$\nabla_{\{x,y\}}(S) = f_{S\cup\{x,y\}}(z_{S\cup\{x,y\}}) - f_{S\cup\{y\}}(z_{S\cup\{y\}}) - [f_{S\cup\{x\}}(z_{S\cup\{x\}}) - f_S(z_S)]. \tag{8}$$

The Shapley interaction index between a feature x and a feature y is split equally between both features, meaning that $\Phi_{\{x,y\}} = \Phi_{\{y,x\}}$. Then, the total interaction effect is $\Phi_{\{x,y\}} + \Phi_{\{y,x\}}$. The main effect can then be computed as the difference between the SHAP value and the total interaction effect for a feature x:

$$\Phi_{\{x,x\}} = \phi_x - \sum_{y \neq x} \Phi_{\{x,y\}}. \tag{9}$$

Given the two applications implemented in the Tree SHAP method and the advantages that come with these applications, we consider the Tree SHAP method for the interpretation method of our decision tree ensemble methods. We use `TreeExplainer` in `python` to compute the SHAP values for the optimal models using the decision tree ensemble methods explained in the previous Sections. For the optimal models we use the optimal parameters obtained from the grid search. We use a summary plot, which orders the variables from the highest feature importance to the lowest feature importance taking into consideration a sample of observations instead of a single observation. Moreover, we conduct the computation of the SHAP values on 100,000 unique random observations. This means for training sets with less than 100,000 observations, the SHAP values are computed for the total training set.

4 Results

In this Section we present the results obtained from the methods explained in Section 3. First, we present the results from the traditional triangular method. Then, we show the results from the grid search and the decision tree ensemble methods. Finally, we give the results from the interpretation method Tree SHAP.

4.1 Mack Chain Ladder

First, we implement the Mack Chain Ladder method, one of the traditional triangular methods, on our data set as explained in Section 3.1. In Table 3 a basic chain ladder is shown for I accident years and J development periods. Filling this table in with the results obtained from the Mack Chain Ladder Method, results in Table 6 as shown below. The light grey area corresponds to the observed accumulated total claim payments for accident year i up until development period j, C_{ij}. This data is then used for the predicted accumulated total claim payments $\hat{C}_{ij}$, denoted as the dark grey area in Table 6.

Table 6: Accumulated total claim payments obtained from the Mack Chain Ladder method.

Accident year i	Development period j											
	0	1	2	3	4	5	6	7	8	9	10	11
1994	36.57	55.80	62.37	65.70	67.59	69.01	69.96	70.74	71.24	71.70	72.08	72.38
1995	35.70	54.63	61.25	64.55	66.45	67.86	68.81	69.58	70.13	70.67	71.11	71.43
1996	35.25	54.77	62.59	66.27	68.49	69.99	71.26	72.09	72.77	73.44	73.79	74.11
1997	34.40	54.07	61.24	64.62	66.60	68.04	69.12	69.98	70.68	71.23	71.69	71.84
1998	35.28	56.28	63.37	67.07	69.23	70.72	71.67	72.53	73.17	73.68	74.13	74.42
1999	36.75	59.39	67.76	71.60	73.88	75.40	76.56	77.35	78.10	78.68	79.18	79.51
2000	36.75	59.27	67.44	71.40	73.64	75.07	76.02	76.68	77.30	77.90	78.40	78.78
2001	38.42	61.33	70.01	74.10	76.46	77.98	79.00	79.78	80.39	80.75	81.08	81.31
2002	38.88	62.72	72.05	76.89	79.63	81.36	82.57	83.49	84.32	84.99	85.47	85.90
2003	44.14	72.89	85.01	90.95	94.34	96.40	97.81	98.84	99.69	100.43	100.93	101.28
2004	42.83	68.21	77.63	82.21	84.88	86.65	87.90	88.78	89.46	90.03	90.46	90.67
2005	46.83	74.94	85.12	89.95	92.77	94.43	95.46	96.37	96.89	97.41	97.82	98.06

Note: This Table shows the result of the Mack Chain Ladder method conducted on our simulated data set. It also corresponds to Table 3 in Section 3.1, but then filled in with available data for this research. The values in this Table are in millions. The light grey area corresponds to the observed values and the dark grey area are to the predicted values with the Mack Chain Ladder method.

When comparing the results in the dark grey area with the observed values in the light grey area, we see that (especially for the earlier development periods) the predicted accumulated total claim payments are higher than the observed accumulated total claim payments for the same development period. However, this can be caused due to the higher value at development period 0 for some accident years and the larger difference in claim payments between development period 1 and development period 0. Since, the Chain Ladder method calculates development factors for the development of the claims, the higher the total claim payments for development period 0, the higher the accumulated total claim payments is in the further development periods.

The resulting claims reserves in millions as calculated in Equation 1 for the Mack Chain Ladder method are shown in the Figure below, next to the actual reserves. The black bar stands for the Ground Truth (GT), which corresponds to the actual reserves and the red bar are the reserves obtained from the Triangular Method (TM). In Figure 8 it can be shown that for the first few accident years, the reserves obtained from the Mack Chain Ladder method are quite similar to the actual reserves. As we predict later accident years the results slightly differ more than in previous accident years. This is caused due to the fact that in later accident years more development periods are predicted and less observed values are used for the prediction. This is also noticeable in Table 6, where for accident year 2005 there are way more predictions made (dark grey cells) than for accident year 1995.

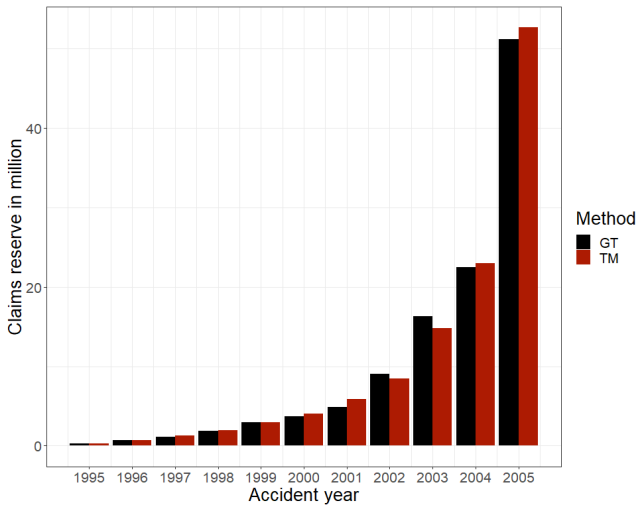


Figure 8: Claims reserves from the Mack Chain Ladder method.

Note: This figure shows the claims reserves per accident year obtained from the Mack Chain Ladder method (TM), next to the actual reserves (GT).

When calculating the Root Mean Squared Error and the percentage difference for each accident year with the predictions of the Mack Chain Ladder method, we get the results as shown in Table 7. In this Table we can also see that for the first few accident years the RMSE is lower than for the last accident years in our data set. This is again due to the more predictions needed for the later accident years. Overall, the RMSE are quite low regarding the high values in claims reserves.

To give a more relative comparison between the different accident years, we look at the percentage difference between the predicted claims reserves obtained from the Mack Chain Ladder and the actual reserves. These results are also shown in Table 7. We notice that for accident year 1999, the relative difference is the smallest among all accident years and for accident year 2001 the largest. Most percentage differences are below the absolute value of 10%.

Table 7: Measures per accident year.

Accident year	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005
RMSE	0.014	0.066	0.108	0.036	0.010	0.370	1.003	0.553	1.437	0.551	1.442
% difference	-4.3%	10.0%	9.3%	1.9%	0.4%	10.0%	20.7%	-6.1%	-8.8%	2.5%	2.8%

Note: This Table shows the Root Mean Squared Error and the percentage difference between the reserves obtained from the Mack Chain Ladder method and the actual reserves. The values in this table for RMSE are also in million.

4.2 Decision tree ensemble methods

First of all, for every decision tree ensemble method we perform hyper parameter tuning as explained in Section 3.3 and shown in Algorithm 1. In Table 8 you can find the full list of optimal parameters used per decision tree ensemble method and per calculation year. Due to the high computation time of the gridsearch, we set a few parameters on fixed values as also explained in Section 3.3. For XGBoost we set the number of rounds (`num_rounds`) on 500, for Random Forest we set the number of trees (`ntree`) on 50 and for the Extra Trees we also set the number of trees (`ntree`) on 50. In Table 8 it can be shown that the parameters for all the three methods we tuned in the algorithm quite differ per

calculation year.

Table 8: Optimal parameter sets resulting from the gridsearch per calculation year.

	XGBoost		Random Forest		Extra Trees		
Calculation year	`max_depth`	`eta`	`mtry`	`nodesize`	`mtry`	`nodesize`	`numRandomCuts`
2006	2	0.5	6	5	6	5	2
2007	2	0.5	6	5	6	3	3
2008	4	0.3	6	3	6	5	2
2009	2	0.3	6	3	6	3	1
2010	2	0.3	6	3	6	5	1
2011	2	0.5	6	3	6	3	1
2012	2	0.3	4	5	6	3	2
2013	8	0.5	4	5	6	3	1
2014	8	0.5	6	3	2	5	1
2015	2	0.5	2	5	2	3	3
2016	4	0.5	4	5	2	3	1

Note: This Table shows the optimal parameters for the decision tree ensemble methods XGBoost, Random Forest and Extra Trees for the estimation of individual claims reserving. All other parameters are set on its default values. For XGBoost we set the number of rounds (`num_rounds`) on 500, for Random Forest we set the number of trees (`ntree`) on 50 and for the Extra Trees we also set the number of trees (`ntree`) on 50.

Then using these optimal parameter sets we calculate the claims reserves for the accident years 1995 till 2005 in our data set as also shown in Algorithm 1. For the calculation of the claims reserves we first need to predict the accumulated total claim payments per claim and per development period. In the Appendix you can find similar Tables as Table 6 but then with the results obtained from the decision tree ensemble methods. The results in these Tables are then used for the calculation of the claims reserves.

The resulting claims reserves in millions as calculated in Equation 1 for the decision tree ensemble methods, XGBoost, Random Forest and Extra Trees, are shown in the Figure below next to the actual reserves and the reserves obtained from the traditional triangular method. Again, the black bar stands for the Ground Truth (GT), which corresponds to the actual reserves, the red bar correspond to the reserves from the traditional triangular method (see Section 4.1), the orange bar are the results obtained from XGBoost (XGB), the yellow bar from the Random Forest (RF) and the pink bar corresponds to the results obtained from the Extra Trees (ET). When comparing the three decision tree ensemble methods with the actual reserves, we notice that the XGBoost algorithm results into claims reserves that are the closest to the actual reserves for almost all accident years. Only for the accident years 2003 and 2004 we can clearly see that the Extra Trees algorithm outperforms the XGBoost and approaches the actual reserves closer.

We also notice in Figure 9, that from the three decision tree ensemble methods the predictions obtained from the XGBoost algorithm approaches the predictions of the traditional triangular method the closest.

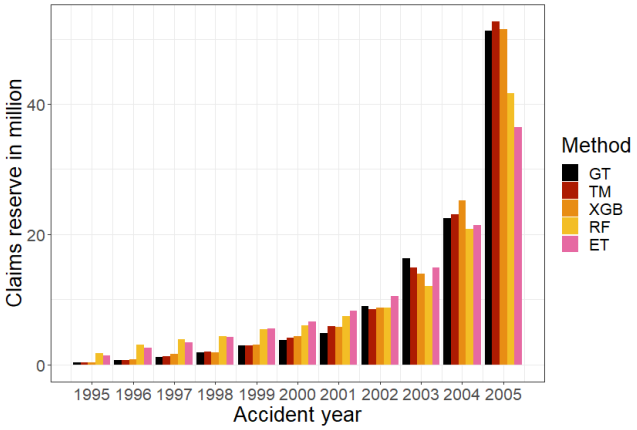


Figure 9: Claims reserves from the decision tree ensemble methods.

Note: This figure shows the claims reserves per accident year obtained from the decision tree ensemble methods XGBoost (XGB), Random Forest (RF) and Extra Trees (ET), next to the actual reserves (GT).

The results shown in the Figure above, are also noticeable in Table 9 below. In this Table it can be seen that for the accident years 2003 and 2004, the RMSE for Extra Trees is smaller than the RMSE for XGBoost and Random Forest. However, for all other accident years the XGBoost algorithm outperforms Random Forest and Extra Trees and the predictions are the closest to the actual reserves. Comparing the results obtained from the decision tree ensemble methods with the triangular method, we observe that the triangular method outperforms all three decision tree ensemble methods for six out of the eleven accident years. But, the RMSE of the triangular method do not differ that much compared to the RMSE of XGBoost.

To give a more relative comparison between the different accident years, we look at the percentage difference between the predicted claims reserves obtained from the decision tree ensemble methods and the actual reserves. These results are also shown in Table 9. We notice the same results as with the comparison with the RMSE. We do observe way higher percentage difference obtained from the Random Forest and Extra Trees algorithm, especially for the first few accident years.

Table 9: RMSE per accident year.

Accident year	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005
RMSE - TM	**0.014**	**0.066**	**0.108**	0.036	**0.010**	**0.370**	1.003	0.553	1.437	**0.551**	1.442
RMSE - XGB	0.020	0.144	0.470	**0.023**	0.103	0.602	**0.911**	0.317	2.374	2.719	**0.243**
RMSE - RF	1.447	2.364	2.730	2.433	2.423	2.341	2.543	**0.285**	4.200	1.629	9.625
RMSE - ET	1.068	1.896	2.284	2.344	2.588	2.857	3.428	1.440	**1.401**	1.072	14.833
% difference - TM	**-4.3%**	**10.0%**	**9.3%**	1.9%	**0.4%**	**10.0%**	20.7%	-6.1%	-8.8%	**2.5%**	2.8%
% difference - XGB	6.2%	21.6%	40.6%	**1.2%**	3.5%	16.2%	**18.8%**	-3.5%	-14.6%	12.1%	**0.5%**
% difference - RF	459.8%	355.4%	235.8%	128.4%	82.1%	63.1%	52.5%	**-3.2%**	-25.8%	-7.3%	-18.8%
% difference - ET	339.5%	285.1%	197.3%	123.7%	86.7%	77.0%	70.7%	16.0%	**-8.6%**	-4.8%	-29.0%

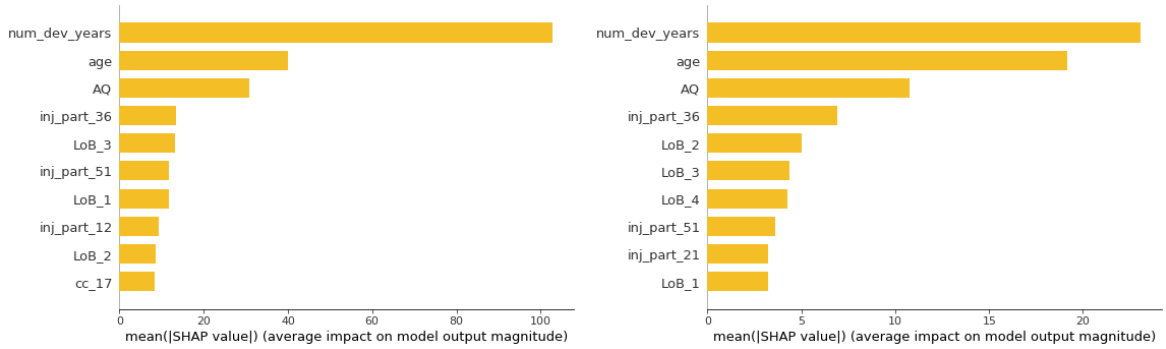
Note: This Table shows the Root Mean Squared Error and the percentage difference between the reserves obtained from the decision tree ensemble methods and the actual reserves. The values in this table are also in million. The values in bold mean that for that specific accident year the corresponding method outperforms the others.

4.3 Tree SHAP

In Section 3.5.1, we explained the interpretation method Tree SHAP for the interpretation of the decision tree ensemble methods. Since we perform eleven models for each of the three decision tree ensemble methods (each accident year has a different model with separate optimal parameters), we use the Tree SHAP method on 33 models (11 accident years $\times$ 3 decision tree ensemble methods).

For each method we show two different summary plots of the SHAP values as shown in Figures 10, 11 and 12. Only the top ten features with the highest impact on the output variable is shown. The two yellow plots in Figure 10 are summary plots obtained from the Random Forest for accident years 1995 and 1997. The two pink plots in Figure 11 are obtained from the Extra Trees algorithm for accident years 1999 and 2001. At last, the last two orange plots in Figure 12 are from the XGBoost algorithm for accident years 2003 and 2005.

We notice in the six plots shown in the Figures below, that the number of development years between the Y.TRAIN and Y.TEST set (`num_dev_years`), accident quarter (`AQ`) and age of the policyholder (`age`) are in all six cases in the top ten of the most important features and for five out of the six cases even in the top five. Only for the last case in Figure 12 (XGBoost for accident year 2005) we notice that the number of development years between the Y.TRAIN and Y.TEST set (`num_dev_years`) is not in the top ten. Also, the injured body parts 36 and 56 are in some cases in the top ten of most important features for the estimation of claims reserving.

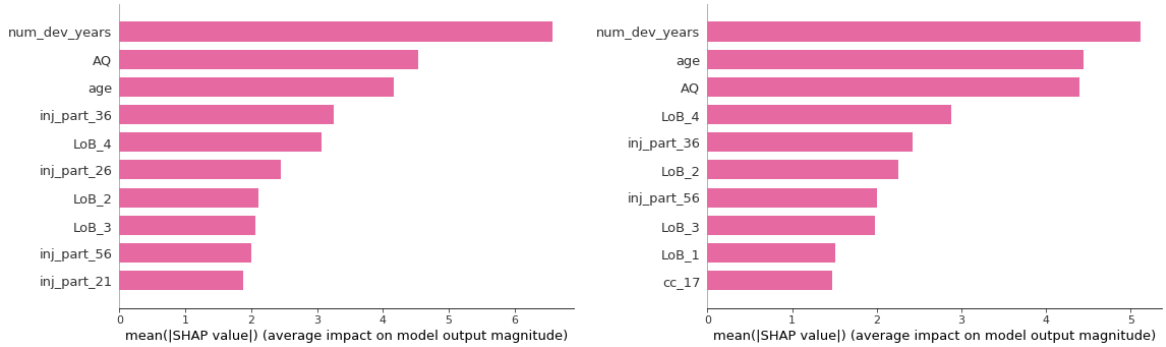


(a) SHAP values for accident year 1995

(b) SHAP values for accident year 1997

Figure 10: SHAP value plots for different accident years obtained from the Random Forest.

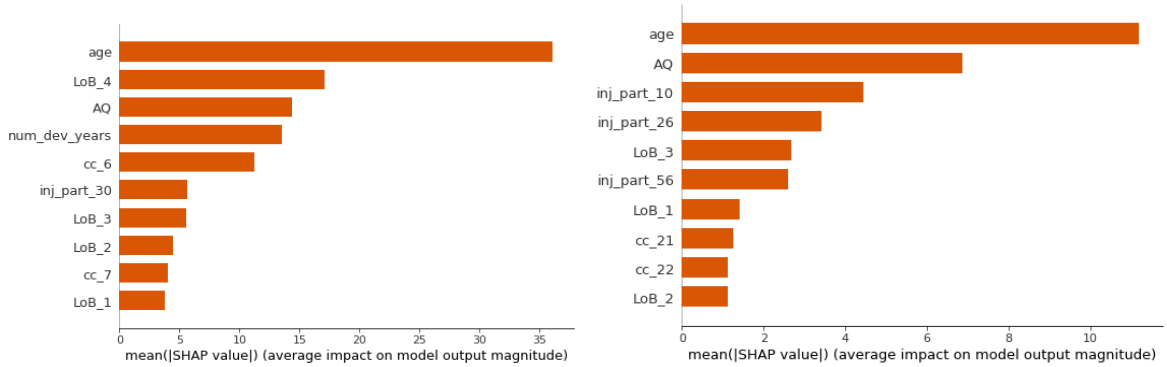
Note: These figures show the global SHAP values. The SHAP values are computed among a sample of 100,000 observations or if the training set consists of less than 100,000 observation, the whole training set. The ten features with the largest impact obtained from the Tree SHAP method are shown in the plots.



(a) SHAP values for accident year 1999 (b) SHAP values for accident year 2001

Figure 11: SHAP value plots for different accident years obtained from Extra Trees.

Note: These figures show the global SHAP values. The SHAP values are computed among a sample of 100,000 observations or if the training set consists of less than 100,000 observation, the whole training set. The ten features with the largest impact obtained from the Tree SHAP method are shown in the plots.



(a) SHAP values for accident year 2003 (b) SHAP values for accident year 2005

Figure 12: SHAP value plots for different accident years obtained from XGBoost.

Note: These figures show the global SHAP values. The SHAP values are computed among a sample of 100,000 observations or if the training set consists of less than 100,000 observation, the whole training set. The ten features with the largest impact obtained from the Tree SHAP method are shown in the plots.

For each method we showed two SHAP plots from two different accident years in Figures 10, 11 and 12. However, we can find similar results for the other cases that we did not incorporate in the Figures. In all the other cases we again find that the number of development years between the Y.TRAIN and Y.TEST set (`num_dev_years`), accident quarter (`AQ`) and age of the policyholder (`age`) are all featured in the top five most important variables.

5 Conclusion

The focus of this research is on the implementation of decision tree ensemble methods for the estimation of individual claims reserving to answer the main research question: "Can decision tree ensemble methods in combination with micro level data outperform the traditional Mack chain ladder method for the estimation of claims reserving? And if so, which method has the best results?".

We used a simulated data set obtained from Gabrielli and V. Wüthrich (2018) to incorporate data on micro level for the estimation of individual claims reserving. We performed the following three decision tree ensemble methods on the data set, namely XGBoost, Random Forest and Extra

Trees. Since, IBNR claims are not known at the time of estimation, we can not use the data of these claims in our estimation process. Therefore, the focus of this research is the estimation of RBNS claims. In Table 1 it can be shown that less than 0.05% of the claims in our data set are IBNR claims. Thus, it is acceptable to focus on the RBNS claims in our research. Then, we recursively build the development of the claims via different training and test sets for each calculation year. The training and test sets are constructed via a splitting criteria explained in Section 3.2. Furthermore, the parameters of each decision tree ensemble method for each calculation year are tuned via a grid search and evaluated with the RMSE. The predictions obtained from the decision tree ensemble methods are then compared with the traditional triangular method, Mack chain ladder. At last, we also performed the interpretation method Tree SHAP on the decision tree ensemble methods, such that we can get to know the underlying factors that play a roll in the estimation of individual claims reserving.

First of all, we conducted the Mack chain ladder method on our simulated data set. The results of this method are explained in Section 4.1. We notice a larger increase between the claims payments in development period 1 and development period 0 than between other development periods. This is not surprising, because in Table 2 we observe the highest mean for the claims paid in the same accident year (`Pay00`). Furthermore, in Table 7 we observe lower RMSE in the first few accident years in comparison with the later accident years. This comes from the fact that the claims are build recursively, so the results of the predictions of the first few accident years are used for the predictions in the later accident years. Overall, the predictions obtained from the Mack chain ladder method are close to the actual reserves with the exception of the predictions in accident year 2001. For accident year 2001 we find a percentage difference of roughly 20% and for all other accident years below an absolute value of 10%. With that being said, we can conclude that the Mack chain ladder predicts the claims reserves well.

Then, to use the decision tree ensemble methods we performed a grid search to find the optimal hyper parameter sets for each decision tree ensemble method per calculation year. In Table 8, we find for every calculation year very different parameter sets for each decision tree ensemble method. Overall, we observe that the amount of features tried at each node (`mtry`) takes in most cases for Random Forest and Extra Trees the highest possible value in the grid, namely 6. This indicates that the trees used in these two decision tree ensemble methods identify the effects that different variables have on the dependent variable. Given the different results in Table 8, we can conclude that the grid search is of importance for the prediction of the claims reserves.

Furthermore, using the optimal hyper parameter sets in Table 8, we conducted the three decision tree ensemble methods. In Figure 9, a comparison of the output of the three decision tree ensemble methods, the Mack chain ladder method and the actual reserves is shown. Among the three decision tree ensemble methods, it can be observed that the XGBoost algorithm comes closest to the actual reserves for almost all accident years with the exception of accident years 2003 and 2004. The Random Forest and Extra Trees algorithm result into a very large percentage difference and also a higher RMSE for the first few accident years. This can be due to the fact that a boosting algorithm, such as XGBoost, helps to reduce the bias and gives therefore more accurate predictions. Also, taking into consideration the computation time of the three different methods, we find that the XGBoost algorithm is significantly faster than the computation time for the Random Forest and Extra Trees algorithm. Overall, it can be concluded that the XGBoost algorithm is the most efficient algorithm

among the three decision tree ensemble methods and gives the most accurate predictions for the individual claims reserves.

However, when comparing the estimates obtained from the XGBoost algorithm with the traditional triangular method, we do find that the traditional triangular method is more accurate than the XGBoost algorithm. In seven out of the eleven accident years the Mack chain ladder method gives more accurate predictions than the XGBoost algorithm. But, the differences are quite small and the XGBoost algorithm still performs very well.

Also, a major disadvantage of using the traditional triangular method is that it does not give any insight in the underlying factors contributing to the prediction of the claims reserves. This is because the traditional triangular methods use data on macro level. However, using data on micro level opens up the possibility to find these underlying factors. This information is very valuable for non-life insurers, since they get to know which characteristics of policyholders might play a big role in the estimation of the claims payments. Also, using machine learning methods in stead of triangular methods make the estimations more robust for changes in the development pattern of the claims.

Moreover, for the interpretation of the decision tree ensemble methods, we use the interpretation method Tree SHAP. Some results of this method are shown in Figures 10, 11 and 12. From these figures we observe that the number of development years between the Y.TRAIN and Y.TEST set (`num_dev_years`), accident quarter (`AQ`) and age of the policyholder (`age`) are almost in all cases in the top five of most important variables for the output. We also sometimes notice the injured body parts 36 and 56 in the top ten. This might indicate that these injured body parts are large injuries with a big impact on the claims payments. Since we do not know the origin of the simulated data set, it is hard to say something about the other variables being in the top ten most important variables.

Overall, after considering all the results obtained from our methodology, the XGBoost algorithm outperforms the three decision tree ensemble methods in its predictions for individual claims reserving. However, it depends on the desires of the insurer to use the traditional triangular method or the XGBoost algorithm for the estimation of claims reserving. Some non-life insurers might find it very important to predict accurate claims reserves and do not care much about the underlying factors that might play a roll in these predictions. For these type of insurers we suggest to stick to the traditional triangular methods, because compared to the decision tree ensemble methods, the traditional triangular methods still give more accurate predictions. On the other hand, other non-life insurers might want to switch from the traditional triangular methods and want to use the available data and computer power that are at its disposal nowadays to get better insight in the data. For them, we suggest to look into using the XGBoost method for the estimation of individual claims reserving. The XGBoost algorithm in combination with data on micro level can give a better understanding in the most important features for the estimation of individual claims reserving with the use of the interpretation method Tree SHAP.

6 Future Research

We have noticed that this research is a hot-topic in the world of non-life insurance. Therefore, further research is needed on the estimation of individual claims reserving.

First of all, we have used simulated data for this research. However, to give better results and

insights in the used methods we recommend to conduct this research on a real life data set from an insurer. This can give a better understanding in the predictions obtained from a certain sector in the non-life insurance industry, such as car insurance, home owners insurance etc. It might be also interesting to compare the results obtained from real life insurance data sets that differ in its origin and to see if XGBoost is still the best performing methods among the decision tree ensemble methods.

Furthermore, the focus of our research is solely on the estimation of RBNS claims. For an insurer the IBNR claims are also very important to estimate, but with the use of our data set we could not have estimated these claims. To be able to estimate the IBNR claims for an insurer, a more complex data set is necessary with more claims information per policyholder. In the simulated data set we used in our research, we have claims that belong to a specific policyholder. However, if it is possible to get data per policy holder, such that we can know the history of each policyholder, it might be possible to estimate the IBNR claims of an insurer. With the use of such kind of data sets the history of claims per policy holder is available and with this information it is possible to calculate the chance of an IBNR claim for a specific policy holder.

Moreover, we have focused on the use of decision tree ensemble methods. We have found in the literature that other machine learning methods, such as neural networks (Wüthrich (2018b)), have also been used on our simulated data set. However, machine learning methods are not the only way to estimate the claims reserves on micro level. It might be interesting to look into parametric ways (Crevecoeur and Antonio (2019a), Pigeon et al. (2013a)) to incorporate the data on micro level and compare the results obtained from parametric methods with non-parametric methods, such as XGBoost.

Altogether, estimating claims reserves on micro level is still a topic that needs to be researched in the future. It is important to keep thinking about ways to incorporate the available data of non-life insurers in their models. Insurers that are interested to make the switch from the traditional triangular methods to more complex models incorporating data on micro level, should keep up with the literature surrounding this subject.

References

E. K. Ampomah, Z. Qin, and G. Nyame. Evaluation of tree-based ensemble machine learning models in predicting stock price direction of movement. *Information*, 11(6), 2020. ISSN 2078-2489. doi: 10.3390/info11060332. URL https://www.mdpi.com/2078-2489/11/6/332.

M. Anthony and P. L. Bartlett. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

K. Antonio and R. Plat. Micro-level stochastic loss reserving for general insurance. *Scandinavian Actuarial Journal*, 2014, 08 2010. doi: 10.2139/ssrn.2111134.

M. Baudry and C. Robert. A machine learning approach for individual claims reserving in insurance. *Applied Stochastic Models in Business and Industry*, 35, 05 2019. doi: 10.1002/asmb.2455.

C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz. A comparative analysis of xgboost. 11 2019.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

J. Chen, L. Song, M. J. Wainwright, and M. I. Jordan. L-shapley and c-shapley: Efficient model interpretation for structured data, 2018.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. pages 785–794, 08 2016. doi: 10.1145/2939672.2939785.

J. Crevecoeur and K. Antonio. A generalized reserving model: bridging the gap between pricing and individual reserving, 10 2019a.

J. Crevecoeur and K. Antonio. A generalized reserving model: bridging the gap between pricing and individual reserving. 10 2019b.

F. Duval and M. Pigeon. Individual loss reserving using a gradient boosting-based approach. *Risks*, 7(3), 2019. ISSN 2227-9091. doi: 10.3390/risks7030079. URL https://www.mdpi.com/2227-9091/7/3/79.

M. Elgedawy. Prediction of breast cancer using random forest, support vector machines and naïve bayes. *International Journal Of Engineering And Computer Science*, 01 2017. doi: 10.18535/ijecs/v6i1.07.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001. doi: 10.1214/aos/1013203451. URL https://doi.org/10.1214/aos/1013203451.

A. Gabrielli and M. V. Wüthrich. An individual claims history simulation machine. *Risks*, 6(2), 2018. ISSN 2227-9091. doi: 10.3390/risks6020029. URL https://www.mdpi.com/2227-9091/6/2/29.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63:3–42, 04 2006. doi: 10.1007/s10994-006-6226-1.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009. URL http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

S. Kotsiantis, D. Kanellopoulos, and P. Pintelas. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30:25–36, 11 2005.

W.-Y. Loh. Fifty years of classification and regression trees. *International Statistical Review / Revue Internationale de Statistique*, 82(3):329–348, 2014. ISSN 03067734, 17515823. URL http://www.jstor.org/stable/43298996.

D. Lubo-Robles, D. Devegowda, V. Jayaram, H. Bedle, K. Marfurt, and M. Pranter. Machine learning model interpretability using shap values: application to a seismic facies classification task. 10 2020. doi: 10.1190/segam2020-3428275.1.

S. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions, 2017.

S. M. Lundberg, G. G. Erion, and S.-I. Lee. Consistent individualized feature attribution for tree ensembles, 2019.

T. Mack. Distribution-free calculation of the standard error of chain ladder reserve estimates. *ASTIN Bulletin*, 23(2): 213–225, 1993. doi: 10.2143/AST.23.2.2005092.

A. Natekin and A. Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 12 2013. doi: 10.3389/fnbot.2013.00021.

A. Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 78, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138385. doi: 10.1145/1015330.1015435. URL https://doi.org/10.1145/1015330.1015435.

B. Pan. Application of xgboost algorithm in hourly pm2.5 concentration prediction. *IOP Conference Series: Earth and Environmental Science*, 113:012127, 02 2018. doi: 10.1088/1755-1315/113/1/012127.

J. Pesantez-Narvaez, M. Guillen, and M. Alcañiz. Predicting motor insurance claims using telematics data—xgboost versus logistic regression. *Risks*, 7(2), 2019. ISSN 2227-9091. doi: 10.3390/risks7020070. URL https://www.mdpi.com/2227-9091/7/2/70.

M. Pigeon, K. Antonio, and M. Denuit. Individual loss reserving with the multivariate skew normal framework. *ASTIN Bulletin*, 43(3):399–428, 2013a. doi: 10.1017/asb.2013.20.

M. Pigeon, K. Antonio, and M. Denuit. Individual loss reserving with the multivariate skew normal framework. *ASTIN Bulletin*, 43(3):399–428, 2013b. doi: 10.1017/asb.2013.20.

M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

P. Schratz, J. Muenchow, E. Iturritxa, J. Richter, and A. Brenning. Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data. *Ecological Modelling*, 406:109–120, 2019. ISSN 0304-3800. doi: https://doi.org/10.1016/j.ecolmodel.2019.06.002. URL https://www.sciencedirect.com/science/article/pii/S0304380019302145.

R. Shafique, A. Mehmood, D. S. Ullah, and G. S. Choi. Cardiovascular disease prediction system using extra trees classifier. 09 2019. doi: 10.21203/rs.2.14454/v1.

L. Shapley. S. 1953."a value for n-person games.". *Contributions to the Theory of Games*, pages 31–40, 1953.

A. Sharaff and H. Gupta. *Extra-Tree Classifier with Metaheuristics Approach for Email Classification*, pages 189–197. 05 2019. ISBN 978-981-13-6860-8. doi: 10.1007/978-981-13-6861-5_17.

D. Ticconi. Individual claims reserving in credit insurance using glm and machine learning. 12 2018. doi: 10.13140/RG.2.2.13118.33600.

G. Valentini and F. Masulli. Ensembles of learning machines. *Neural Nets WIRN Vietri-2002, Series Lecture Notes in Computer Sciences*, 2486:3–22, 05 2002. doi: 10.1007/3-540-45808-5_1.

M. Wüthrich. Machine learning in individual claims reserving. *Scandinavian Actuarial Journal*, 2018:1–16, 01 2018a. doi: 10.1080/03461238.2018.1428681.

M. Wüthrich. Neural networks applied to chain–ladder reserving. *European Actuarial Journal*, 8, 10 2018b. doi: 10.1007/s13385-018-0184-4.

L. Yang and A. Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, Nov 2020. ISSN 0925-2312. doi: 10.1016/j.neucom.2020.07.061. URL http://dx.doi.org/10.1016/j.neucom.2020.07.061.

X. Zhang, Y. Yang, and Z. Zhou. A novel credit scoring model based on optimized random forest. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 60–65, 2018. doi: 10.1109/CCWC.2018.8301707.

X. Zhao and X. Zhou. Applying copula models to individual claim loss reserving methods. *Insurance: Mathematics and Economics*, 46(2):290–299, April 2010. URL https://ideas.repec.org/a/eee/insuma/v46y2010i2p290-299.html.

J. Zhou and J. Garrido. A loss reserving method based on generalized linear models. 06 2009. doi: 10.13140/RG.2.2.31371.23844.

A Appendix

Construction of a training and test set

Table 10: Construction of training and test sets for $t^* = 2007$.

	Development year										
Accident year i	2000	2001	2002	2003	2004	2005	2006	**2007**	2008	2009	2010
	Observed C_{ij}						Predicted $\hat{C}_{ij}$				
2000 ($i=1$)	C_{10}	C_{11}	..	..	..	C_{15}	×	×	×	×	×
2001 ($i=2$)	×	C_{20}	C_{21}	..	..	..	$\hat{C}_{25}$	×	×	×	×
2002 ($i=3$)	×	×	C_{30}	C_{31}	..	..	..	$\hat{C}_{35}$	×	×	×
2003 ($i=4$)	×	×	×	C_{40}	C_{41}	..	..	..	$\hat{C}_{45}$	×	×
2004 ($i=5$)	×	×	×	×	C_{50}	C_{51}	..	..	..	$\hat{C}_{55}$	×
2005 ($i=6$)	×	×	×	×	×	C_{60}	$\hat{C}_{61}$	..	..	..	$\hat{C}_{65}$

	Development year										
Accident year i	2000	2001	2002	2003	2004	2005	2006	**2007**	2008	2009	2010
	Observed C_{ij}						Predicted $\hat{C}_{ij}$				
2000 ($i=1$)	C_{10}	C_{11}	..	..	..	C_{15}	×	×	×	×	×
2001 ($i=2$)	×	C_{20}	C_{21}	..	..	..	$\hat{C}_{25}$	×	×	×	×
2002 ($i=3$)	×	×	C_{30}	C_{31}	..	..	..	$\hat{C}_{35}$	×	×	×
2003 ($i=4$)	×	×	×	C_{40}	C_{41}	..	..	..	$\hat{C}_{45}$	×	×
2004 ($i=5$)	×	×	×	×	C_{50}	C_{51}	..	..	..	$\hat{C}_{55}$	×
2005 ($i=6$)	×	×	×	×	×	C_{60}	$\hat{C}_{61}$	..	..	..	$\hat{C}_{65}$

Note: This table shows the construction of the training and test sets for the calculation time $t^* = 2007$. The yellow area (X.TRAIN) denotes the data we use for the training of the model. The orange area (Y.TRAIN) is the dependent variable we use for the training of the model. The light blue is (X.TEST) the data used in the trained model for predicting the accumulated total claim payments in 2007, denoted as the dark blue area Y.TEST).

Algorithms for the decision tree ensemble methods

Algorithm 2 Gradient boosting algorithm.

Input: input data $(x, y)_{i=1}^{N}$, number of iterations M, choice of loss function $\Psi(y, f)$, choice of base-learner model $h(x, \theta)$

1: Initialize $\hat{f}_0$ with a constant
2: **for** $t = 1$ till M **do**
3: Compute the gradient $g_t(x)$
4: Fit a new base-learner function $h(x, \theta_t)$
5: Find the best gradient descent step-size ρ_t: $\rho_t = \arg\min_\rho \sum_{i=1}^{n} \Psi[y_i, \hat{f}_{t-1}(x_i) + \rho h(x_i, \theta_t)]$
6: Update the function estimate: $\hat{f}_t \leftarrow \hat{f}_{t-1} + \rho_t h(x, \theta_t)$
7: **end for**

Note: This algorithm shows the gradient boosting algorithm in short. We use as base-learners decision trees and for the choice of loss function the sum of squared error. The number of iterations M is set to 1000. Note that this algorithm shows the gradient boosting algorithm, but in our research we use the XGBoost algorithm (see Section 3.4.1 for the differences). Source by: Natekin and Knoll (2013).

Algorithm 3 Random Forest algorithm for regression purposes.

Input: input data $(x, y)_{i=1}^{N}$, number of trees B, bootstrap sample size K, number of variables used as candidates at each split m, minimum node size n_{min}

Output: Ensemble of trees $\{T_b\}_1^B$ with predictions $\{y_b\}_1^B$

1: **for** $b = 1$ till B **do**
2: Draw a bootstrap sample of size K from the training data.
3: Grow a regression tree T_b to the bootstrapped data.
4: (i) Select m variables at random from the total of p variables.
5: (ii) Pick the best splitting point among the m variables.
6: (iii) Split the node into two 'daughter' nodes.
7: Repeat the above steps until the minimum node size n_{min} is reached.
8: **end for**
9: To get the Random Forest prediction: $\hat{y} = B^{-1} \sum_{b=1}^{B} y_b$

Note: This algorithm shows the Random Forest algorithm for regression purposes. We tune the number of trees B, number of variables used as candidates at each split m and the minimum node size n_{min} with hyper parameter tuning as explained in Section 3.3. Source by: Breiman (2001)

Results of the reserved obtained from the decision tree ensemble methods

Table 11: Accumulated total claim payments obtained from XGBoost.

Accident year i	Development period j											
	0	1	2	3	4	5	6	7	8	9	10	11
1994	36.57	55.80	62.37	65.70	67.59	69.01	69.96	70.74	71.24	71.70	72.08	72.38
1995	35.70	54.63	61.25	64.55	66.45	67.86	68.81	69.58	70.13	70.67	71.11	71.48
1996	35.25	54.77	62.59	66.27	68.49	69.99	71.26	72.09	72.77	73.44	73.95	74.25
1997	34.40	54.07	61.24	64.62	66.60	68.04	69.12	69.98	70.68	71.42	71.96	72.31
1998	35.28	56.28	63.37	67.07	69.23	70.72	71.67	72.53	73.11	73.71	74.16	74.45
1999	36.75	59.39	67.76	71.60	73.88	75.40	76.56	77.47	78.15	78.85	79.31	79.61
2000	36.75	59.27	67.44	71.40	73.64	75.07	76.30	77.20	77.92	78.61	79.09	79.38
2001	38.42	61.33	70.01	74.10	76.46	77.86	79.08	79.97	80.68	81.40	81.90	82.22
2002	38.88	62.72	72.05	76.89	79.30	80.72	82.11	83.03	83.75	84.45	84.95	85.59
2003	44.14	72.89	85.01	89.93	92.43	93.99	95.21	96.14	96.93	97.67	98.24	98.90
2004	42.83	68.21	80.04	84.82	87.23	88.70	89.90	90.83	91.59	92.46	93.11	93.38
2005	46.83	72.53	84.63	89.68	92.08	93.57	94.74	95.68	96.78	97.55	98.00	98.31

Note: This Table shows the result of XGBoost conducted on our simulated data set. It also corresponds to Table 3 in Section 3.1, but then filled in with available data for this research. The values in this Table are in millions. The light grey area corresponds to the observed values and the dark grey area are to the predicted values with the XGBoost algorithm.

Table 12: Accumulated total claim payments obtained from the Random Forest.

Accident year i	Development period j											
	0	1	2	3	4	5	6	7	8	9	10	11
1994	36.57	55.80	62.37	65.70	67.59	69.01	69.96	70.74	71.24	71.70	72.08	72.38
1995	35.70	54.63	61.25	64.55	66.45	67.86	68.81	69.58	70.13	70.67	71.11	72.88
1996	35.25	54.77	62.59	66.27	68.49	69.99	71.26	72.09	72.77	73.44	75.23	76.47
1997	34.40	54.07	61.24	64.62	66.60	68.04	69.12	69.98	70.68	72.56	73.82	74.57
1998	35.28	56.28	63.37	67.07	69.23	70.72	71.67	72.53	74.28	75.47	76.18	76.86
1999	36.75	59.39	67.76	71.60	73.88	75.40	76.56	78.48	79.84	80.65	81.36	81.93
2000	36.75	59.27	67.44	71.40	73.64	75.07	77.05	78.40	79.25	79.99	80.54	81.12
2001	38.42	61.33	70.01	74.10	76.46	78.75	80.24	81.20	82.10	82.75	83.32	83.85
2002	38.88	62.72	72.05	76.89	79.58	81.28	82.43	83.32	84.01	84.62	85.16	85.62
2003	44.14	72.89	85.01	88.94	91.45	92.77	93.85	94.78	95.48	96.16	96.64	97.08
2004	42.83	68.21	77.61	81.58	83.85	85.08	86.16	87.00	87.66	88.24	88.66	89.04
2005	46.83	67.64	77.28	81.18	83.20	84.45	85.51	86.36	87.10	87.69	88.12	88.44

Note: This Table shows the result of the Random Forest method conducted on our simulated data set. It also corresponds to Table 3 in Section 3.1, but then filled in with available data for this research. The values in this Table are in millions. The light grey area corresponds to the observed values and the dark grey area are to the predicted values with the Random Forest algorithm.

Table 13: Accumulated total claim payments obtained from Extra Trees.

Accident year i	Development period j											
	0	1	2	3	4	5	6	7	8	9	10	11
1994	36.57	55.80	62.37	65.70	67.59	69.01	69.96	70.74	71.24	71.70	72.08	72.38
1995	35.70	54.63	61.25	64.55	66.45	67.86	68.81	69.58	70.13	70.67	71.11	72.50
1996	35.25	54.77	62.59	66.27	68.49	69.99	71.26	72.09	72.77	73.44	74.91	76.01
1997	34.40	54.07	61.24	64.62	66.60	68.04	69.12	69.98	70.68	72.27	73.41	74.13
1998	35.28	56.28	63.37	67.07	69.23	70.72	71.67	72.53	74.24	75.39	76.17	76.77
1999	36.75	59.39	67.76	71.60	73.88	75.40	76.56	78.60	79.90	80.77	81.47	82.07
2000	36.75	59.27	67.44	71.40	73.64	75.07	77.36	78.75	79.68	80.47	81.12	81.64
2001	38.42	61.33	70.01	74.10	76.46	79.27	80.91	82.02	82.94	83.66	84.27	84.74
2002	38.88	62.72	72.05	76.89	80.51	82.61	83.94	84.92	85.69	86.36	86.91	87.34
2003	44.14	72.89	85.01	90.91	93.80	95.45	96.59	97.51	98.26	98.89	99.40	99.88
2004	42.83	68.21	77.85	82.06	84.22	85.51	86.55	87.42	88.11	88.72	89.21	89.59
2005	46.83	65.31	73.29	76.72	78.38	79.53	80.48	81.27	81.95	82.47	82.90	83.23

Note: This Table shows the result of Extra Trees conducted on our simulated data set. It also corresponds to Table 3 in Section 3.1, but then filled in with available data for this research. The values in this Table are in millions. The light grey area corresponds to the observed values and the dark grey area are to the predicted values with the Extra Trees algorithm.