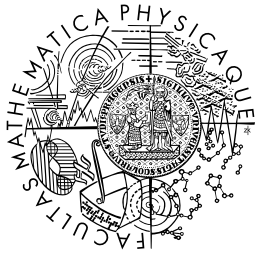


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Jan Janoušek

Bagging and regression trees in individual claims reserving

Department of Probability and Mathematical Statistics

Supervisor of the master thesis: doc. RNDr. Michal Pešta, Ph.D.

Study programme: Financial and Insurance Mathematics

Study branch: Financial and insurance mathematics

Prague 2023

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

Prague, May 3, 2023 .....

Jan Janoušek

I want to thank my supervisor, *doc. RNDr. Michal Pešta, Ph.D.*, for his guidance, valuable advice, endless support, and almost instant responses to all of my e-mails.

I would also like to thank my immediate family, namely my mom *Jana*, my dad *Radim* and my little brother *Jonáš* together with my girlfriend *Míša* for their continuous support and care during my whole university studies. Lastly, I would like to thank my late grandfather *děda Otik* for his unstoppable support and for never stopping to believe in me during our time together.

Title: Bagging and regression trees in individual claims reserving

Author: Jan Janoušek

Department: Department of Probability and Mathematical Statistics

Supervisor: doc. RNDr. Michal Pešta, Ph.D., department

Abstract: This diploma thesis focuses on the application of classification and regression trees, as well as bootstrap aggregating, to individual reserving in insurance. In the first part, we provide a summary of the theory and establish mathematical formalities that are sometimes overlooked in basic texts on these topics. We provide a comprehensive overview of the concepts, including a detailed discussion of their practical applications. In the second part, we build on existing research by extending the use of machine learning in individual claims reserving. Specifically, we expand on a prior article that only modeled the number of claims using classification trees. We also incorporate regression trees and bagging to model the size of each claim, resulting in more accurate reserve estimates. We achieve this by applying these techniques to insurance data and obtaining empirical distributions that allow us to calculate confidence intervals and quantiles. Ultimately, we determine the reserves needed for both the next year and the ultimate reserves.

Keywords: bagging regression tree claims reserving micro reserving bootstrap non-life insurance

Contents

Introduction

In this diploma thesis, we will first introduce classification and regression trees and carefully build the theory involving them. Then we will briefly introduce bootstrap aggregating. In the second part of the thesis, we will introduce the problem of insurance claims reserving and apply the regression trees and bootstrap aggregating techniques to the claim-by-claim reserving problem on simulated data. Claim-by-claim reserving technique is also known as individual reserving, micro reserving, or granular reserving.

Classification and regression trees are important tools in the field of machine learning, which is concerned with the development of algorithms and models that can learn patterns and make predictions from data. One of the key challenges in machine learning is to develop models that can accurately predict outcomes based on input data. Classification and regression trees are useful in this regard because they can handle both categorical and continuous data and can handle interactions between different features or variables.

A popular modification of classification and regression trees is bootstrap aggregating or better known as bagging. Bagging uses bootstrapped datasets to build many different classification or regression trees and then averages the results. We will also briefly introduce this concept and apply both classification and regression trees together with bagging on the insurance claim reserving.

Our task will be to apply the theory from classification and regression trees together with bagging in the field of insurance reserving, namely on a claim-by-claim basis. The main idea is to model the reserves for each claim individually instead of modeling the reserves for aggregates. This is done in the second part of the thesis. We will apply both classification and regression trees on simulated data and make predictions for the next year's reserves and also for the ultimate reserves.

The first theoretical part of the thesis will be based on Breiman et al. [1993]. The second, more practical part of this thesis will be based on Wüthrich [2018], and we will try to elaborate on this article. We will try to extend the results from that article to predict not only the number of payments but also the size of the payments. Thanks to the bagging approach, we will have an empirical distribution that allows us to calculate such things as confidence intervals, means, standard deviations, and quantiles. This will also be used to calculate the Solvency Capital Requirement that insurance companies need to fulfill according to Solvency II.

1. Classification and regression trees

1.1 Introduction to classification and regression trees

In this chapter, we will build classification and regression trees theory. Classification and regression trees are supervised machine learning algorithms that, in each step, recursively partition the data into two subsets based on the outcome of a question until some stopping criterion is reached. Answering the consecutive questions then decides what to predict from new observations. This algorithm is shortly called CART.

We will try to illustrate most of the terms defined in a simple illustrative example regarding the risk of heart disease. We will have two covariates, age (in years) and weight (in kilograms), and one response variable risk with three classes: low risk, medium risk, and high risk. As we build the theory, we will illustrate the defined terms in this simple example.

1.2 Classification

1.2.1 Classifiers, partitions and learning samples

We will begin our discussion with the topic of classification, covering classifiers in general before delving into trees and their specific use in classification. In this section, we will introduce fundamental terms such as feature space, vector of covariates, and response variables.

Definition 1 (Feature space, vector of covariates and response variables)
Let $\mathbf{X} = (X_1, X_2, \ldots, X_M)^\top$ *be a random vector with values in a general* M*-dimensional space* $\mathcal{X} = \mathcal{X}^1 \times \mathcal{X}^2 \times \ldots \times \mathcal{X}^M$*. We will call* $\mathbf{X}$ *a* feature vector *or* vector of covariates *and* $\mathcal{X}$ *a* feature space.

Moreover, let $J \in \{2, 3, \ldots\}$*, i.e.,* $J \in \mathbb{N} \setminus \{1\}$*, and let* Y *be a categorical random variable with values in the set* $\{1, 2, \ldots, J\}$*. Variable* Y *will be called* response variable*. The pair* $(\mathbf{X}, Y)^\top$ *will be called feature-response pair.*

Finally, we will define the probability space. Let $\Omega = \mathcal{X} \times \{1, 2, \ldots, J\} = \mathcal{X}^1 \times \mathcal{X}^2 \times \ldots \times \mathcal{X}^M \times \{1, 2, \ldots, J\}$.

If $\mathbf{X}_i, i \in \{1, 2, \ldots, J\}$ *is continuous, then its sigma-algebra* $\mathfrak{X}_i$ *is defined as the Borel sigma-algebra* $\mathcal{B}(\mathcal{X}^i)$*. On the other hand if* $\mathbf{X}_i, i \in \{1, 2, \ldots, J\}$ *is categorical or nominal, then its sigma-algebra* $\mathfrak{X}_i$ *is defined as the powerset* $2^{\mathcal{X}^i}$ *(or sometimes denoted by* $\mathcal{P}(\mathcal{X}^i)$*). Then the sigma-algebra on* Ω *is defined as the product sigma-algebra*

$$\mathcal{F} = \mathfrak{X}_1 \otimes \mathfrak{X}_2 \otimes \ldots \otimes \mathfrak{X}_M \otimes 2^{\{1,2,\ldots,J\}} =$$
$$= \sigma\Big(\{A_1 \times A_2 \times \ldots \times A_M \times B \colon A_i \in \mathfrak{X}_i,\ i \in \{1, 2, \ldots, J\},\ B \subseteq \{1, 2, \ldots, J\}\}\Big)$$

Lastly, the probability $\mathbb{P}$ *is a mapping* $\mathbb{P} \colon \mathcal{F} \to [0, 1]$ *that satisfies the axioms of probability measure, i.e.,*

1. $\mathbb{P}(F) \geq 0$, *for each* $F \in \mathcal{F}$

2. $\mathbb{P}(\Omega) = 1$

3. $\mathbb{P}\left(\bigcup_{i=1}^{+\infty} F_i\right) = \sum_{i=1}^{+\infty} \mathbb{P}(F_i)$ *for all pairwise disjoint* $F_1, F_2, \ldots \in \mathcal{F}$.

Example. In our illustrative example, the feature space is two-dimensional, and it holds that $\mathcal{X} = \mathcal{X}^1 \times \mathcal{X}^2$, where $\mathcal{X}^1 = [0, 100]$ and $\mathcal{X}^2 = [0, 300]$. An example of a vector of covariates may have realization $\mathbf{X} = (35, 80)^\top$, i.e., an observation of a 35-year-old weighing 80 kilograms.

$J = 3$ in our example, and naturally, we would code low risk as 1, medium risk as 2, and high risk as 3. △

Remark. In the rest of this thesis, it is assumed that J belongs to the set of natural numbers greater than or equal to 2, i.e., $J \in \{2, 3, \ldots\}$. This assumption will be implicitly understood and not repeated △

Our task now will be to assign a value from the set $\{1, 2, \ldots, J\}$ to each feature vector $\mathcal{X}$. This process will be called *classifying.*

Definition 2 (Classifier)
Let $\mathcal{X}$ *be a feature space and* Y *be a response variable with* J *possible classes* $\{1, 2, \ldots, J\}$. *A function* $d\colon \mathcal{X} \to \{1, 2, \ldots, J\}$ *is called a* classifier *if the mapping* $d\colon \mathbf{X} \mapsto d(\mathbf{X})$ *assigns for each* $\mathbf{X} \in \mathcal{X}$ *exactly one element of the set* $\{1, 2, \ldots, J\}$.

Example. An example (although very naive) of a classifier in our illustrative example would be a classifier that would classify each vector of covariates as medium risk, i.e., $d(\mathbf{X}) = 2$, for each $\mathbf{X} \in \mathcal{X}$. This classifier is probably a very *bad* one. However, we do not know yet what even *bad* means for a classifier. We will define ways to measure the goodness of a classifier later. △

We can view the mapping of the classifier as assigning a single element from the set $\{1, 2, \ldots, J\}$ to each vector of covariates $\mathbf{X}$ in the feature space $\mathcal{X}$. This perspective allows us to see that the feature space $\mathcal{X}$ is partitioned into J regions based on the output of the classifier. We refer to this partitioning as the *partition* of $\mathcal{X}$ and the resulting regions as *partitions.* The following definition summarizes this approach.

Definition 3 (Partitioning feature space into regions)
Let $\mathcal{X}$ *be a feature space. We will say that* $\mathcal{X}$ *was* partitioned into J regions $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_J$, $\mathcal{X}_i \neq \emptyset$, *if* $\mathcal{X} = \bigcup_{i=1}^{J} \mathcal{X}_i$ *and* $\mathcal{X}_i \bigcap \mathcal{X}_j = \emptyset$ *for any* $i, j \in \{1, 2, \ldots, J\}$, $i \neq j$, *i.e., if* $\mathcal{X}$ *can be represented as a disjoint union of nonempty sets* $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_J$.

The reader may notice that the definition of partitioning feature space into regions does not depend on the classifier. However, we will think of classifiers as a mathematical representation of partitioning.

Example. In our illustrative example, it could be the case that the feature space $\mathcal{X}$ is partitioned into three regions $\mathcal{X}_1 = \{(a, w) \in [0, 30] \times [0, 80]\}$, i.e., those people with weight less than or 80 kilograms and younger or 30-year-old. $\mathcal{X}_2 = \{(a, w) \in ([0, 65] \times [0, 100]) \setminus ([0, 30] \times [0, 80])\}$, i.e., those people who are either

older than 30 and younger or 65 years old, or weight more than 80 and less than or 100 kilograms. Finally, $\mathcal{X}_3 = \{(a, w) \in ([0, 100] \times [0, 300]) \setminus ([0, 65] \times [0, 100])\}$, i.e., those people who are older than 65 (and younger than or 100) or who weight more than 100 kilograms (and less than or 300). The partitioning is depicted in Figure 1.1. △

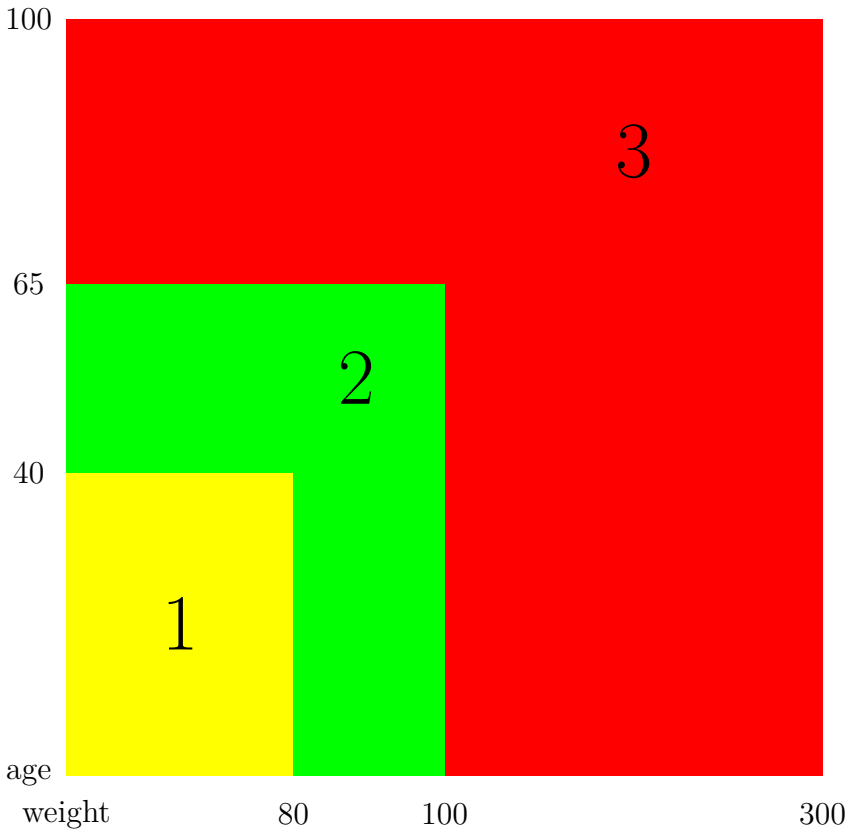


Figure 1.1: An example of the partitioning of the feature space.

So far, we have discussed classifiers, feature spaces, partitions, and regions. However, all these terms would be useless if we did not have any data. For this reason, we will introduce the term *learning sample* in the following definition.

Definition 4 (Learning sample)
Let $\mathcal{X}$ be a feature space. The collection of N feature-response pairs $(\mathbf{X}^1, Y^1)^\top$, $(\mathbf{X}^2, Y^2), \ldots, (\mathbf{X}^N, Y^N)$, where $\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^N \in \mathcal{X}$ and $Y^1, Y^2, \ldots, Y^N \in \{1, 2, \ldots, J\}$ will be called a learning sample *or* learning dataset*, and will be denoted by*

$$\mathcal{L} = \left\{\left(\mathbf{X}^1, Y^1\right)^\top, \left(\mathbf{X}^2, Y^2\right)^\top, \ldots, \left(\mathbf{X}^N, Y^N\right)^\top\right\}.$$

Sometimes we refer to it as training sample/dataset.

Example. An example of a learning sample with 6 observations may be

$$\mathcal{L} = \left\{ \left((30, 100)^\top, 1\right)^\top, \left((20, 90)^\top, 1\right)^\top, \left((50, 110)^\top, 3\right)^\top, \right.$$
$$\left. \left((48, 91)^\top, 1\right)^\top, \left((56, 130)^\top, 2\right)^\top, \left((78, 170)^\top, 3\right)^\top \right\}$$

△

As we mentioned before, we need some way to measure the goodness of a classifier. We will cover this in the next subsection. We will define the *true misclassification rate* and some ways to estimate it.

1.2.2 Goodness of a classifier

We have defined a classifier (at the moment, we do not care about how the classifier assigns classes or how *good* or *bad* the classifier is).

As we mentioned earlier, an example of a classifier could be mapping $d\colon \mathcal{X} \to \{1, 2, \ldots, J\}$ such that for all $\mathbf{X} \in \mathcal{X}\colon d(\mathbf{X}) = 2$. It is obvious that such a classifier cannot stand in general. However, we need some way to measure the goodness of a classifier. This was done in the following definition.

Definition 5 (True misclassification rate)
Let $\mathcal{L}$ be a learning sample, d be a classifier on the feature space $\mathcal{X}$ with J possible classes. Also let $(\mathbf{X}, Y)^\top$ be a feature-response pair, where $\mathbf{X}$ is from $\mathcal{X}$ and Y is a response variable with J classes, drawn randomly from the same probability distribution

$$P(A, j) = \mathbb{P}(\mathbf{X} \in A, Y = j), A \subseteq \mathcal{X}, j \in \{1, 2, \ldots, J\}$$

independent of $\mathcal{L}$.

Then we will call the quantity

$$R^*(d) = \mathbb{P}(d(\mathbf{X}) \neq Y)$$

a true misclassification rate.

During the evaluation of the probability $\mathbb{P}(d(\mathbf{X}) \neq Y)$, we consider the learning sample $\mathcal{L}$ to be fixed. Thus a more precise notation would be

$$\mathbb{P}(d(\mathbf{X}) \neq Y \mid \mathcal{L}),$$

i.e., the probability of misclassifying the new sample $(\mathbf{X}, Y)^\top$ given the learning sample $\mathcal{L}$.

The true misclassification rate is a theoretical quantity that needs to be somehow estimated. We will incorporate three methods of estimation similar to, for example, simple linear regression.

The resubstitution estimate

Let $\mathcal{L}$ be the learning sample of size N and d be classifier that was built based on the learning sample $\mathcal{L}$.

The resubstitution estimate of the classifier d based on the learning data $\mathcal{L}$ will be denoted by $R(d)$ and is calculated by the following formula.

$$R(d) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(d(\mathbf{X}_i) \neq Y_i),$$

i.e., it calculates the proportion of misclassified observations from the learning sample $\mathcal{L}$. ($\mathbb{I}(x)$ denotes an indicator function, i.e., it takes the value of 1 of the statement x is true and 0 otherwise.)

The drawback of the resubstitution estimate is that it uses the same data as it used for constructing the classifier d instead of an independent sample. Thus it could give an overly optimistic estimation of the accuracy of d.

Test sample estimation

The second method is based on randomly dividing the learning sample $\mathcal{L}$ into two sets: $\mathcal{L}_1$ and $\mathcal{L}_2$ of sizes N_1 and N_2, where $\mathcal{L} = \mathcal{L}_1 \bigcup \mathcal{L}_2$ and $\mathcal{L}_1 \bigcap \mathcal{L}_2 = \emptyset$ (and thus $N_1 + N_2 = N$).

We use only the sample $\mathcal{L}_1$ to construct the classifier d and then we use the sample $\mathcal{L}_2$ to estimate $R^*(d)$. The test sample estimate (denoted by $R^{ts}(d)$) is calculated via:

$$R^{ts}(d) = \frac{1}{N_2} \sum_{\{i:(x_i,y_i)\in\mathcal{L}_2\}} \mathbb{I}(d(\mathbf{X}_i) \neq Y_i),$$

i.e., it calculates the proportion of misclassified observations from the test sample $\mathcal{L}_2$. This method eliminates the drawback of the resubstitution method as it uses different data for constructing the estimate and for calculating its accuracy, but there arises another problem as we only use a part of the data for the construction of d while the rest of the data had to be left out while constructing d so that it can be used independently in testing. This problem may be neglected when large datasets are used. In these cases, it is also preferred as it is much less computationally demanding than the last method which is *cross validation.*

Cross validation

This method is called K-fold cross-validation, and it randomly divides all data in $\mathcal{L}$ into K subsets of similar sizes (sizes of different subsets differ at most by one). Let us denote these subsets by $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_K$.

Now for $k \in \{1, 2, \ldots, K\}$, the method uses all subsets apart from the k-th one, that is $\mathcal{L} \setminus \mathcal{L}_k$ as a learning sample (this was denoted by $\mathcal{L}_1$ in the previous method-Test sample estimation) and $\mathcal{L}_k$ as $\mathcal{L}_2$. It calculates test sample estimate of k-th group denoted by $R^{ts}(d^{(k)})$.

Then it does the same for all $k \in \{1, 2, \ldots, K\}$ and calculates the cross-validation estimate $R^{CV}(d)$ as

$$R^{CV}(d) = \frac{1}{K} \sum_{k=1}^{K} R^{ts}(d^{(k)}),$$

i.e., it takes the average over all $k \in \{1, 2, \ldots, K\}$. The most popular modifications of cross-validations are leave-one-out cross-validation, i.e., we choose $K = N$, the size of the learning sample $\mathcal{L}$, and 10-fold cross-validation (where we choose $K = 10$). This method combines the advantages of both previously mentioned methods. Its drawback is that it is really computationally expensive to calculate, especially for the leave-one-out cross-validation. Thus this method is used when only small datasets are at hand, as it both eliminates the drawback of the test sample method of using only a part of the data and is not so computationally expensive as we have only a small dataset. If we have large datasets, it is preferred to use the test sample estimation.

Example. Let us assume that the classifier d partitioned the feature space $\mathcal{X}$ into three regions depicted in Figure 1.1, i.e., into those regions described in the example at the end of section 1.3.1. And let us assume that it partitioned the feature space based on the learning sample $\mathcal{L}$ described in the example after

Definition 4. We will now describe how the resubstitution estimate is calculated:

$$R(d) = \frac{1}{6}\Big(\mathbb{I}\left(d\left((30,100)^\top\right) \neq 1\right) + \mathbb{I}\left(d\left((20,90)^\top\right) \neq 1\right) + \mathbb{I}\left(d\left((50,110)^\top\right) \neq 3\right)$$
$$+ \mathbb{I}\left(d\left((48,91)^\top\right) \neq 1\right) + \mathbb{I}\left(d\left((56,130)^\top\right) \neq 2\right) + \mathbb{I}\left(d\left((78,170)^\top\right) \neq 3\right)\Big)$$
$$= \frac{1}{6}\left(\mathbb{I}\left(2 \neq 1\right) + \mathbb{I}\left(2 \neq 1\right) + \mathbb{I}\left(3 \neq 3\right) + \mathbb{I}\left(2 \neq 1\right) + \mathbb{I}\left(3 \neq 2\right) + \mathbb{I}\left(3 \neq 3\right)\right) = \frac{2}{3}.$$

Thus based on our (very limited) learning sample, the resubstitution estimate for the true misclassification rate is $\frac{2}{3}$.

Describing the test sample and cross-validation estimation would be lengthy and not so instructive. Thus, we will limit ourselves to only showing the resubstitution estimate as an example. △

1.3 Tree classifiers

So far, we have discussed only classifiers in general. Now, we will have a look at a particular class of classifiers called binary tree-structured classifiers. As the name suggests, the classifiers have something in common with binary trees. We will start by defining binary trees and some terms regarding binary trees that we will need in the future.

Definition 6 (A binary tree)
A binary tree *is a set* $\mathcal{T}$ *of natural numbers containing at least the number* 1, *together with a mapping* $son\colon \mathcal{T} \to \mathcal{T}$ *such that the mapping* $son\colon t \mapsto son(t)$, *satisfies:*

- *for each* $t \in \mathcal{T}$ *it either holds that* $son(t) = \emptyset$ *or* $son(t) = \{2t, 2t+1\}$
- *for each* $s \in \mathcal{T} \setminus \{1\}$ *there exists exactly one* $t \in \mathcal{T}$ *such that* $s \in son(t)$.

We will call all elements of the set $\mathcal{T}$ nodes. *The node numbered* 1 *will be called the* root node *and nodes* $t \in \mathcal{T}$ *for which* $son(t) = \emptyset$ *will be called* terminal nodes *or* leaves. *The set of all terminal nodes of tree* $\mathcal{T}$ *will be denoted by* $\tilde{\mathcal{T}}$.

For $a, b \in \mathcal{T}$, *if* $a \in son(b)$ *then we say that* a *is a* son node (or son) *of* b *and that* b *is a* father node (or father) *of* a.

For $o, p \in \mathcal{T}$, *if there exist a* $U \in \mathbb{N}$ *and a sequence* $a_1, a_2, \ldots, a_U$ *of elements of* $\mathcal{T}$ *satisfying:*

- a_1 *is a son of* o
- a_{i+1} *is a son of* a_i *for* $i \in \{1, 2, \ldots, U-1\}$
- p *is a son of* a_U,

then we will call o *an* ancestor of p. *And if* o *is an ancestor of* p, *then* p *will be called a descendant of* o.

We will call the set $S \subseteq \mathbb{N}$ *a* subtree *of* $\mathcal{T}$ *if*

- $S \subseteq \mathcal{T}$
- S *is a binary tree.*

Let $t \in \mathcal{T}$. The subtree consisting of the node t and all its descendats will be called branch *and will be denoted by $\mathcal{T}_t$. The root node of the tree $\mathcal{T}_t$ is the node t.*

In the rest of the thesis, every time we mention a tree, it is meant as a binary tree.

A binary tree is usually graphically displayed as in Figure 1.2 with root up and leaves down. In the figure, we can see a tree with 11 nodes, and 6 of them are terminal. They are the nodes numbered: $5, 7, 8, 9, 12, 13$.

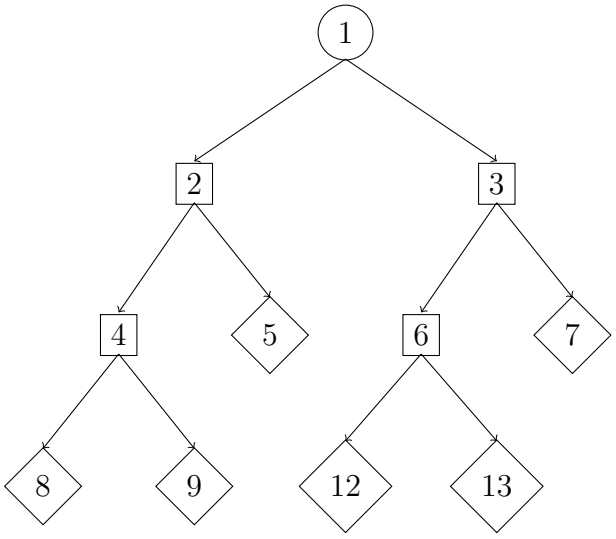


Figure 1.2: An example of a binary tree (root is depicted in a circle, nodes in a square and terminal nodes in diamonds).

Now that we have defined a binary tree, we will move to classifiers that can be represented by a binary tree, i.e., binary tree-structured classifiers. However, before defining binary tree-structured classifiers, we need to define *standardized set of questions* that will be used in the definition of the binary tree-structured classifier.

Definition 7 (Standardized set of questions)
We say that a question Q is from the standardized set of questions *denoted by $\mathcal{Q}$ if it follows:*

1. *Each split depends on the value of a single variable;*
2. *For each numerical and ordered variable X_m, $m \in \{1, 2, \ldots, M\}$ the set $\mathcal{Q}$ includes all questions of the form $\{Is\ X_m \leq c?\}$ and $\{Is\ X_m \geq c?\}$ for $c \in \mathbb{R}$;*
3. *For each categorical and nominal variable $X_m, m \in \{1, 2, \ldots, M\}$ taking values from the set $\{b_1, b_2, \ldots, b_O, O \in \mathbb{N}\}$ the set $\mathcal{Q}$ includes all questions of the form $\{Is\ X_m \in S?\}$ for S ranging over all subsets $\{b_1, b_2, \ldots, b_O, O \in \mathbb{N}\}$.*

Example. In our illustrative example, a splitting question from the standardized set of questions $\mathcal{Q}$ could be, for example: $\{\text{Is } X_1 \leq 100?\}$ (This is exactly the way we will split the feature space in 1.3.1).

An example of a question that does not belong to the standardized set of questions $\mathcal{Q}$ could be, for example, $\{\text{Is } X_1 \leq 100, \text{ or is } X_2 \leq 20?\}$, because it depends on two variables. Also, a question $\{\text{Is } 0 \leq X_1 < 5?\}$ is not allowed as it has to be split into two questions. $\triangle$

Now we can finally define binary tree-structured classifiers.

Definition 8 (Binary tree-structured classifier)
Let $\mathcal{L}$ be a learning sample. A classifier d built based on the learning sample $\mathcal{L}$ and constructed by repeatedly partitioning subsets of the feature space $\mathcal{X}$ (that will be denoted $\mathcal{X}_1$) into two descendant subsets will be called binary tree-structured classifier.

If we partition a part of the feature space $\mathcal{X}$ numbered $k \in \mathbb{N}$, then the two descendant subsets will be numbered $2k$ and $2k+1$. Thus, we can write that $\mathcal{X}_{2k} \bigcup \mathcal{X}_{2k+1} = \mathcal{X}_k$ and $\mathcal{X}_{2k} \bigcap \mathcal{X}_{2k+1} = \emptyset$.

Those subsets which are not partitioned anymore are called terminal subsets. *The terminal subsets form a partitioning of the feature space $\mathcal{X}$. Each terminal subset is assigned a class label (we will discuss later how it is done). It is possible that two or more terminal subsets have the same class label. The partition into regions corresponding to the classifier d is achieved by a union of all terminal subsets corresponding to the same class. The partitions are formed by conditions on the coordinates of the covariates vector $\mathbf{X}$.*

The binary tree-structured classifier predicts a class for each vector of covariates $\mathbf{X}$ in the following way: We look at the questions laid upon the vector of covariates $\mathbf{X}$ and depending on the outcome of the question we decide to which subset it belongs to. The questions have to be from the standardized set of questions $\mathcal{Q}$. When the vector of covariates reaches the terminal subset, its predicted class is given by the class label assigned to that terminal subset.

By size of a region $\mathcal{X}_t$, *we will denote the number of observations from the training sample $\mathcal{L}$ contained in $\mathcal{X}_t$.*

Thanks to the standardized set of questions $\mathcal{Q}$, if the vector of covariates $\mathbf{X}$ consists of only ordinal and numerical variables, we can look at the decision tree as a procedure that recursively partitions the feature space into (hyper-)rectangles with sides parallel to the axes. This is illustrated in Figure 1.3.

The partitioning of a feature space $\mathcal{X}_1$ into terminal subsets satisfies the Definition 3 of partitioning into regions. The numbering that $\mathcal{X}_k$ is partitioned into $\mathcal{X}_{2k}$ and $\mathcal{X}_{2k+1}$ was chosen so that it corresponds to the terminology of binary trees, i.e., sons of node k are nodes $2k$ and $2k+1$.

Now that we have defined binary trees and binary tree-structured classifiers, we can switch our terminology to the more natural one adapted from the binary trees.

Definition 9 (Tree theory terminology)
Additionally, we will use a slightly different notation:

- *Each subset of the feature space $\mathcal{X}$ will be named* node.
  Specifically for $u \in \mathbb{N}$, the subset $\mathcal{X}_u$ will be denoted by node number u.
- *The whole feature space $\mathcal{X}$ also denoted by $\mathcal{X}_1$ will be called* root node.

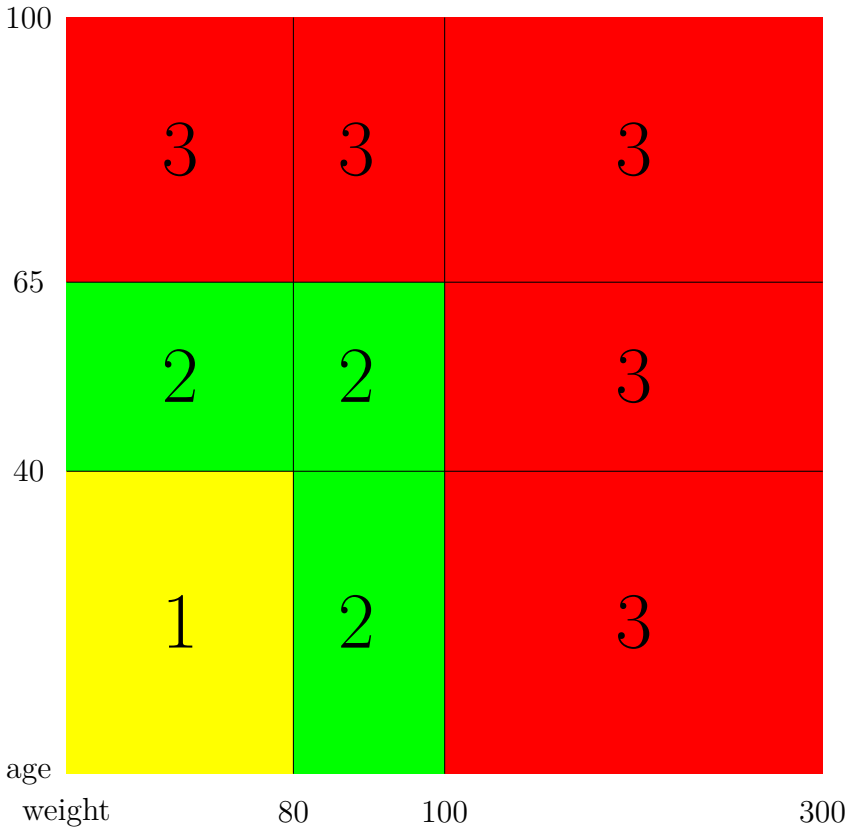


Figure 1.3: An example of the partitioning of the feature space.

- *Terminal subsets will be called* terminal nodes.
- *Partitioning a subset* $\mathcal{X}_l$, $l \in \mathbb{N}$ *of the feature space* $\mathcal{X}$ *will be called* splitting of node l *and will be precisely defined in the following definition.*

Now that we have defined binary tree-structured classifiers, we can switch in the notation from d as a classifier to $\mathcal{T}$ as a binary tree-structured classifier.

Lastly, in this subsection, we will look more closely at what a *split* means in the following definition.

Definition 10 (Split)
Let $\mathcal{T}$ *be a binary tree-structured classifier and* $t \in \mathcal{T}$. *We say that the node* t *is split by a* split s *into two son nodes* $2t$ *(also called* t_L *as a left son) and* $2t+1$ *(also called* t_R *as a right son) if the subset* $\mathcal{X}_t$ *of the feature space* $\mathcal{X}$ *was partitioned into regions* $\mathcal{X}_{2t}$ *and* $\mathcal{X}_{2t+1}$.

The split s *is made upon an answer to a question* Q *regarding the vector of covariates from the standardized set of questions* $\mathcal{Q}$. *Usually, the part of observations answering by "yes" will become a part of the left son node* (t_L), *while the others answering by "no" will become a part of the right son node* (t_R). *However, this is not considered a strict rule, and it does not have any impact on the outcome.*

Construction of a tree

In this section, we will have a look at the construction of a binary tree-structured classifier. To construct a tree, we need to find answers to the three following questions:

- How to select a split?
- What class to assign to a terminal node?
- When to stop splitting?

We will start with the first question.

1.3.1 How to select a split?

Let us look at the first problem, which means finding binary splits of the root node 1 into smaller nodes based on the learning sample $\mathcal{L}$. The intuitive idea is to make each of the son nodes more homogeneous than the father node.

Example. In our illustrative example, it would mean that for the first split, we could choose to split the data into two parts based on the weight of the observed persons. If they weigh over 100 kilograms, we would divide them into the first group and, if less, into the second group.

That means we would have in node 2 the following observations:

$$\left((50,110)^\top,3\right)^\top,\left((56,130)^\top,2\right)^\top,\left((78,170)^\top,3\right)^\top$$

and in node 3 observations:

$$\left((30,100)^\top,1\right)^\top,\left((20,90)^\top,1\right)^\top,\left((48,91)^\top,1\right)^\top.$$

Let us call this split s. △

Once a good split of node 1 is found, we look at both son nodes 2 and 3 independently and try to find a good split for them, and we continue iteratively to nodes of smaller and smaller sizes.

Let us now have a look at how to measure the homogeneity mentioned in the previous paragraphs. We will define something called *impurity.* For that, we need to define node proportions. As the name suggests, it is the proportion of observations of a given class j in a given node t.

Definition 11 (Node proportions)
Let $\mathcal{T}$ be a binary tree-structured classifier with J classes, $t \in \mathcal{T}$, $j \in \{1,2,\ldots,J\}$ and $\left((\mathbf{X}_1,Y_1)^\top,(\mathbf{X}_2,Y_2)^\top,\ldots,(\mathbf{X}_N,Y_N)^\top\right)$ be a vector of feaure-response pairs, where $\mathbf{X}_i \in \mathcal{X}$, $i \in \{1,2,\ldots,N\}$, and $Y_i \in \{1,2,\ldots,J\}$, $i \in \{1,2,\ldots,N\}$.

We define the $p(j \mid t)$ as the proportion of covariate vectors $\mathbf{X}_1,\mathbf{X}_2,\ldots,\mathbf{X}_N$ from t belonging to class $j \in \{1,2,\ldots,J\}$, *i.e., $p(\cdot \mid t)$ is a mapping $p(\cdot \mid t)\colon J \to [0,1]$*

$$p(j \mid t) = \frac{\sum_{i=1}^{N} \mathbb{I}(\mathbf{X}_i \in t)\mathbb{I}(Y_i = j)}{\sum_{i=1}^{N} \mathbb{I}(\mathbf{X}_i \in t)}.$$

The quantity $\sum_{i=1}^{N} \mathbb{I}(\mathbf{X}_i \in t)$ *denotes the* number of observations in node t *and will be denoted by* $N(t)$.

The quantity $\sum_{i=1}^{N} \mathbb{I}(Y_i = j)$ *denotes the* number of observations in the whole tree $\mathcal{T}$ with class j *and will be denoted by* N_j.

The quantity $\sum_{i=1}^{N} \mathbb{I}(\mathbf{X}_i \in t)\mathbb{I}(Y_i = j)$ *denotes* the number of observations in node t belonging to class j *and it will be denoted by* $N_j(t)$.

Example. To continue with our illustrative example: the node proportions for node 2 are: $p(1 \mid 2) = 0$, $p(2 \mid 2) = \frac{1}{3}$ and $p(3 \mid 2) = \frac{2}{3}$ and for node 3 $p(1 \mid 3) = 1$ and $p(2 \mid 3) = p(3 \mid 3) = 0$.

The number of observations in node 2 and 3 is in our example following: $N(2) = N(3) = 3$. The number of observation with classes 1, 2, and 3 is $N_1 = 3$, $N_2 = 1$, $N_3 = 2$. The number of observations in nodes 2 and 3 with classes 1, 2, and 3 are: $N_1(2) = 0, N_1(3) = 3, N_2(2) = 1, N_2(3) = 0, N_3(2) = 2, N_3(3) = 0$. $\triangle$

It is easy to see that for each $t \in \mathcal{T}$ that $\sum_{j=1}^{J} p(j \mid t) = 1$. Now that we have defined node proportions, we can define the *impurity.*

Definition 12 (Impurity)
Let $\mathcal{T}$ *be a binary tree-structured classifier with* J *classes and* $t \in \mathcal{T}$. *We define* impurity *of node* t *denoted by*

$$i(t) = \varphi(p(1 \mid t), p(2 \mid t), \ldots, p(J \mid t)),$$

where φ *is a non-negative function*

$$\varphi\colon [0,1]^J \to [0, +\infty), \;\; \varphi\colon u_1, u_2, \ldots u_J \mapsto \varphi(u_1, u_2, \ldots u_J)$$

that satisfies the following assumptions:

- φ *is maximized if and only if the proportions are spread equally over all classes, i.e.,*

$$\underset{(u_1,u_2,\ldots,u_J)\in[0,1]^J}{\operatorname{argmax}} \varphi(u_1, u_2, \ldots, u_J) = \left\{ \left(\frac{1}{J}, \frac{1}{J}, \ldots, \frac{1}{J}\right) \right\}.$$

- φ *is symmetric, i.e.,*

$$\varphi(u_1, u_2, \ldots, u_J) = \varphi\left(u_{\sigma(1)}, u_{\sigma(2)}, \ldots, u_{\sigma(J)}\right)$$

  for each permutation σ *on the set* $\{1, 2, \ldots, J\}$.

- φ *takes its minimal value of* 0 *if and only if only one class is present, i.e., if*

$$\begin{aligned} \varphi(1, 0, \ldots, 0) = \varphi(0, 1, \ldots, 0) &= \varphi(0, \ldots, 0, 1, 0, \ldots, 0) \\ = \varphi(0, \ldots, 1, 0) &= \varphi(0, \ldots, 0, 1) = 0, \end{aligned}$$

  and if $a, b > 0$, *then* $\varphi(a, b, 0, \ldots, 0) > 0$.

We will now define two ways how to measure impurity: *Gini index* and *cross-entropy.*

Definition 13 (Gini index)
Let $\mathcal{T}$ be a binary tree-structured classifier with J classes and t a node within $\mathcal{T}$ with class proportions $p(1 \mid t), p(2 \mid t), \ldots, p(J \mid t)$, then we will call

$$G(t) = \sum_{j=1}^{J} p(j \mid t)(1 - p(j \mid t))$$

a Gini index *of node t.*

Theorem 1 (Gini index is an impurity)
Let $\mathcal{T}$ be a binary tree-structured classifier with J classes, $t \in \mathcal{T}$, and $G(t)$ be the Gini index of node t, then the Gini index is an impurity as defined in Definition 12.

Proof. Let $J \in \mathbb{N}$ be fixed and let u_i denote $p(i \mid t)$. We need to show that for $u_1, u_2, \ldots, u_J \in [0, 1]$, such that $\sum_{j=1}^{J} u_j = 1$ it holds that

$$\varphi_G\left(u_1, u_2, \ldots, u_J\right) = \sum_{j=1}^{J} u_j(1 - u_j)$$

is non-negative and:

- $\varphi_G(u_1, u_2, \ldots, u_J)$ is maximized if and only if $u_1 = u_2 = \ldots = u_J = \frac{1}{J}$.
- $\varphi_G(u_1, u_2, \ldots, u_J) = \varphi_G(u_{\sigma(1)}, u_{\sigma(2)}, \ldots, u_{\sigma(J)})$ for any permutation σ on the set $\{1, 2, \ldots, J\}$.
- $\varphi_G(u_1, u_2, \ldots, u_J)$ takes its minimal value of 0 if an only if $u_j = 1$, $j \in \{1, 2, \ldots, J\}$ (consequently $u_1, u_2, \ldots, u_{j-1}, u_{j+1}, \ldots, u_J = 0$).

The proof of non-negativity is obvious as $u_i(1 - u_i)$ is non-negative for $u_i \in [0, 1]$, and the sum of non-negative numbers is also non-negative.

The second point is easy to see as

$$\begin{aligned}\varphi_G(u_1, u_2, \ldots, u_J) &= \sum_{j=1}^{J} u_j(1 - u_j) = \sum_{j=1}^{J} u_{\sigma(j)}(1 - u_{\sigma(j)}) \\ &= \varphi_G(u_{\sigma(1)}, u_{\sigma(2)}, \ldots, u_{\sigma(J)}),\end{aligned}$$

where the second equality is obvious, as the order of summation does not matter.

For the first point, We will divide the set $[0, 1]^J$ into the open set $(0, 1)^J$ and the border $[0, 1]^J \setminus (0, 1)^J$. We prove the first part using the Lagrange multiplier technique. We will denote the function we want to find extremes for by f and the function denoting the constraints by g.

Thus we need to find extremes of function

$$f(u_1, u_2, \ldots u_J) = u_1(1 - u_1) + u_2(1 - u_2) + \ldots + u_J(1 - u_J)$$

on the open set $(0,1)^J$. We will take the constraint $u_1 + u_2 + \ldots + u_J = 1$ as function g such that

$$g(u_1, u_2, \ldots u_J) = u_1 + u_2 + \ldots + u_J - 1 = 0.$$

Then the Lagrange function ϕ is:

$$\begin{aligned}\phi(u_1, u_2, \ldots, u_J, \lambda) &= f(u_1, u_2, \ldots, u_J) + \lambda g(u_1, u_2, \ldots, u_J) \\ &= u_1(1-u_1) + u_2(1-u_2) + \ldots + u_J(1-u_J) + \lambda(u_1 + u_2 + \ldots + u_J - 1).\end{aligned}$$

Thanks to the symmetry, it holds for the partial derivatives with respect to u_i, where $i \in \{1, 2, \ldots, J\}$ that

$$\frac{\partial}{\partial u_i}\phi(u_1, u_2, \ldots, u_J, \lambda) = 1 - 2u_i + \lambda,$$

thus if we set it to equal to zero, we get $u_i = \frac{1+\lambda}{2}$ for $i \in \{1, 2, \ldots, J\}$ and for the partial derivative with respect to λ, it holds that

$$\frac{\partial}{\partial \lambda}\phi(u_1, u_2, \ldots, u_J, \lambda) = u_1 + u_2 + \ldots + u_J - 1.$$

If we also set the second derivative to be equal to zero, we get back to the constraint that $u_1 + u_2 + \ldots + u_J = 1$.

By plugging the derivatives with respect to u_i's into the derivative with respect to λ, we get $J\frac{1+\lambda}{2} = 1$ from which we get that $\lambda = \frac{2}{J} - 1$ and finally by plugging λ back into the expressions for u_i we get that for $i \in \{1, 2, \ldots, J\}$ it holds that

$$u_i = \frac{1+\lambda}{2} = \frac{1 + \frac{2}{J} - 1}{2} = \frac{1}{J}.$$

If we take the second partial derivative of f, we get that

$$\frac{\partial^2}{\partial u_i \partial u_j} f(u_1, u_2, \ldots, u_J) = -2\mathbb{I}(i = j).$$

Thus it is easy to see that for the Hessian matrix $\mathbb{H}$ it holds that

$$\mathbb{H} = \operatorname{diag}\Big\{-2, -2, \ldots, -2\Big\},$$

therefore the matrix has only negative eigenvalues, and thus the matrix is negative definite, and the point $\left(\frac{1}{J}, \frac{1}{J}, \ldots, \frac{1}{J}\right)$ is the only point of local maximum.

For the values in the set $[0,1]^J \setminus (0,1)^J$, i.e., in the set $\{u_i \in \{0,1\}, i \in \{1, 2, \ldots, J\}\}$ it is easy to see that all the numbers in the sum are equal to zero as either u_i or $1 - u_i$ is equal to zero. If we also incorporate the constraint that $u_1 + u_2 + \ldots + u_J = 1$, we get that only one of the components is equal to 1, and the others have to be zero. This proves the third point and, thus, the whole theorem. □

Example. Let us continue with our motivation example. We will calculate the Gini index of nodes 1, 2, and 3.

$$\begin{aligned}
G(1) &= \sum_{j=1}^{J} p(j \mid 1)\,(1 - p(j \mid 1)) = p(1 \mid 1)\,(1 - p(1 \mid 1)) \\
&\quad + p(2 \mid 1)\,(1 - p(2 \mid 1)) + p(3 \mid 1)\,(1 - p(3 \mid 1)) \\
&= \frac{3}{6}\left(1 - \frac{3}{6}\right) + \frac{1}{6}\left(1 - \frac{1}{6}\right) + \frac{2}{6}\left(1 - \frac{2}{6}\right) = \frac{9}{36} + \frac{5}{36} + \frac{8}{36} = \frac{22}{36} = \frac{11}{18}. \\
G(2) &= \sum_{j=1}^{J} p(j \mid 2)\,(1 - p(j \mid 2)) = p(1 \mid 2)\,(1 - p(1 \mid 2)) \\
&\quad + p(2 \mid 2)\,(1 - p(2 \mid 2)) + p(3 \mid 2)\,(1 - p(3 \mid 2)) \\
&= 0\,(1 - 0) + \frac{1}{3}\left(1 - \frac{1}{3}\right) + \frac{2}{3}\left(1 - \frac{2}{3}\right) = 0 + \frac{2}{9} + \frac{2}{9} = \frac{4}{9}.
\end{aligned}$$

And because node 3 is pure, $G(3) = 0$. △

Definition 14 (Cross entropy)
Let $\mathcal{T}$ be a binary tree-structured classifier with J classes and t a node within $\mathcal{T}$ with class proportions $p(1 \mid t), p(2 \mid t), \ldots, p(J \mid t)$, then we will call

$$D(t) = -\sum_{j=1}^{J} p(j \mid t) \log_2(p(j \mid t)).$$

a cross entropy *of node t.*

In the following theorem and its proof, we will frequently use the identity $\log_2(x) = \frac{\log(x)}{\log(2)}$ for $x \in (0, +\infty)$. If we write simply $\log(x)$ we mean the natural logarithm, i.e., $\log_e(x) = \ln(x)$. We will also use notation $\log(0)$ which means $\lim_{x \to 0+} \log(x)$ and consequently common limit known from the basic courses of analysis: $\lim_{x \to 0+} x \log(x) = 0$.

Theorem 2 (Entropy is an impurity)
Let $\mathcal{T}$ be a binary tree-structured classifier with J classes, $t \in \mathcal{T}$, and $D(t)$ be the Cross-entropy of node t, then the Cross-entropy is an impurity as defined in Definition 12.

Proof. Let $J \in \mathbb{N}$ be fixed and let u_i denote $p(i \mid t)$. We need to show that for $u_1, u_2, \ldots, u_J \in [0, 1]$, such that $\sum_{j=1}^{J} u_j = 1$ it holds that

$$\varphi_E(u_1, u_2, \ldots, u_J) = -\sum_{j=1}^{J} u_j \log_2(u_j)$$

is non-negative and:

- $\varphi_E(u_1, u_2, \ldots, u_J)$ is maximized if and only if $u_1 = u_2 = \ldots = u_J = \frac{1}{J}$.
- $\varphi_E(u_1, u_2, \ldots, u_J) = \varphi_E(u_{\sigma(1)}, u_{\sigma(2)}, \ldots, u_{\sigma(J)})$ for any permutation σ on the set $\{1, 2, \ldots, J\}$.

- $\varphi_E(u_1, u_2, \ldots, u_J)$ takes its minimal value of 0 if an only if $u_j = 1,\ j \in \{1, 2, \ldots, J\}$ (consequently $u_1, u_2, \ldots, u_{j-1}, u_{j+1}, \ldots, u_J = 0$).

The proof of non-negativity is obvious as $-u_i \log_2(u_i)$ is non-negative for $u_i \in [0, 1]$, as $-\log_2(u_i)$ is non-negative, and the sum of non-negative numbers is also non-negative.

The second point is easy to see as

$$\begin{aligned}\varphi_E(u_1, u_2, \ldots, u_J) &= -\sum_{j=1}^{J} u_j \log_2(u_j) = -\sum_{j=1}^{J} u_{\sigma(j)} \log_2(u_{\sigma(j)}) \\ &= \varphi_E(u_{\sigma(1)}, u_{\sigma(2)}, \ldots, u_{\sigma(J)}),\end{aligned}$$

where the second equality is again obvious, as the order of summation does not matter.

We will prove the second point again by splitting $[0, 1]^J$ into two sets $(0, 1)^J$ and $[0, 1]^J \setminus (0, 1)^J$ and the first part again by the Lagrange multiplier technique. We need to find extremes of function

$$f(u_1, u_2, \ldots u_J) = -\left(u_1 \log_2(u_1) + u_2 \log_2(u_2) + \ldots + u_J \log_2(u_J)\right)$$

on the open set $(0, 1)^J$. We will take the constraint $u_1 + u_2 + \ldots + u_J = 1$ again as function g such that $g(u_1, u_2, \ldots u_J) = u_1 + u_2 + \ldots + u_J - 1 = 0$. Then the Lagrange function ϕ is:

$$\begin{aligned}&\phi(u_1, u_2, \ldots, u_J, \lambda) = f(u_1, u_2, \ldots, u_J) + \lambda g(u_1, u_2, \ldots, u_J) \\ &= -(u_1 \log_2(u_1) + u_2 \log_2(u_2) + \ldots + u_J \log_2(u_J)) + \lambda\left(u_1 + u_2 + \ldots + u_J - 1\right)\end{aligned}$$

Thanks to the symmetry, it holds for the partial derivatives with respect to u_i, where $i \in \{1, 2, \ldots, J\}$ that

$$\frac{\partial}{\partial u_i}\phi(u_1, u_2, \ldots, u_J, \lambda) = \frac{-1}{\log(2)}(\log(u_i) + 1) + \lambda,$$

thus if we set it to equal to zero, we get

$$u_i = \exp\{-1 + \log(2)\lambda\} = \frac{2^\lambda}{e}$$

and for the partial derivative with respect to λ, it holds that

$$\frac{\partial}{\partial \lambda}\phi(u_1, u_2, \ldots, u_J, \lambda) = u_1 + u_2 + \ldots + u_J - 1.$$

If we also set the second derivative to be equal to zero, we get back to the constraint that $u_1 + u_2 + \ldots + u_J = 1$.

By plugging the derivatives with respect to u_i's into the derivative with respect to λ, we get $J\frac{2^\lambda}{e} = 1$ from which we get that $\lambda = \log_2\left(\frac{e}{J}\right)$ and finally by plugging λ back into the expressions for u_i we get that

$$u_i = \frac{2^\lambda}{e} = \frac{\frac{e}{J}}{e} = \frac{1}{J}.$$

If we take the second partial derivative of f, we get that

$$\frac{\partial^2}{\partial u_i \partial u_j} f(u_1, u_2, \ldots, u_J) = \frac{-\mathbb{I}(i=j)}{\log(2) u_i}.$$

Thus it is easy to see that for the Hessian matrix $\mathbb{H}$ it holds that

$$\mathbb{H} = \operatorname{diag}\left\{\frac{-1}{\log(2)u_1}, \frac{-1}{\log(2)u_2}, \ldots, \frac{-1}{\log(2)u_J}\right\}.$$

Thus the matrix has only negative eigenvalues, and thus the matrix is negative definite, and the point $\left(\frac{1}{J}, \frac{1}{J}, \ldots, \frac{1}{J}\right)$ is the only point of local maximum.

For the values in the set $[0,1]^J \setminus (0,1)^J$, i.e., in the set $\{u_i \in \{0,1\}, i \in \{1,2,\ldots,J\}\}$ it is easy to see that all the numbers in the sum are equal to zero as $u_i \log_2(u_i)$ is equal to zero. If we also incorporate the constraint that $u_1 + u_2 + \ldots + u_J = 1$, we get that only one of the components is equal to 1, and the others have to be zero. This proves the third point and, thus, the whole theorem.

□

The Gini index and Cross entropy graphs for the case when there are only two classes ($J = 2$) are depicted in Figure 1.4. We can see that it does not differ very much. This corresponds to the result that it does not really matter which one of these two we choose. In part regarding how to stop splitting, we will see that the choice of stopping criterion is much more important than the choice between the Gini index and Cross entropy.

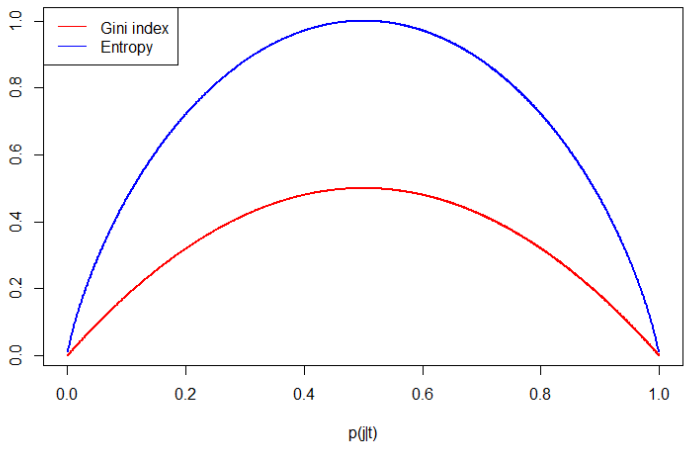


Figure 1.4: Graph of Gini index and cross-entropy for $J = 2$.

Decrease in impurity

What we want split s to do is to decrease impurity as much as possible. Thus we define the *decrease in impurity.*

Definition 15 (Decrease in impurity)
Let $t \in \mathcal{T}$ and s be a split that divides node t into nodes t_L and t_R such that a proportion p_{t_L} of cases in node t goes into node t_L and proportion p_{t_R} of cases in node t goes into node t_R

We say that the split s of node t has a decrease in impurity *of*

$$\Delta i(s,t) = i(t) - (p_{t_L} i(t_L) + p_{t_R} i(t_R)) .$$

Example. Now we can measure the decrease in the impurity of split s we described in Section 1.3 of node 1. We will use the Gini index to measure impurity.

$$\Delta G(s,1) = G(1) - (p_2 G(2) + p_3 G(3)) = \frac{11}{18} - \left(\frac{1}{2}\cdot\frac{4}{9} + \frac{1}{2}\cdot 0\right) = \frac{11}{18} - \frac{4}{18} = \frac{7}{18}.$$

△

As we already mentioned, we want the decrease in impurity to be as large as possible. Thus we will always choose a split that yields the highest possible decrease in impurity.

If we now jump to the question of when to stop splitting. We can come up with the idea that we could stop splitting when the decrease in impurity does not exceed some threshold $\beta \geq 0$. We will discuss in the later part of the thesis why this is not a very successful approach.

Definition 16 (Probability of being in a given node and/or belonging to a given class)
The probability of a new feature-response pair $(\mathbf{X}, Y)^\top$, randomly drawn from the same distribution as was the learning sample $\mathcal{L}$, to be in node t and belonging to class j will be denoted by $P(j,t)$.

The probability of a new feature response pair $(\mathbf{X}, Y)^\top$, randomly drawn from the same distribution as was the learning sample $\mathcal{L}$, to be in node t denoted by $P(t)$.

And finally, the probability of a new data-line $(\mathbf{X}, Y)^\top$, randomly drawn from the same distribution as was the learning sample $\mathcal{L}$, to belong to class j given that it is in node t is denoted by $P(j \mid t)$.

For the quantities we have just defined, we have the following resubstitution estimates:

$$p(j,t) = \frac{N_j(t)}{N}.$$

$P(t)$ can be estimated by

$$p(t) = \sum_{j=1}^{J} p(j,t) = \frac{N_t}{N}.$$

$P(j \mid t)$ can be estimated by

$$p(j \mid t) = \frac{p(j,t)}{p(t)},$$

we have seen this earlier in the definition of node proportions (definition 11).

Example. In our illustrative example, it holds for example that $p(1,3) = \frac{1}{2}$ as there are three observation in the node 3 and all of them are of class 1, $p(3) = \frac{1}{2}$ as there are three observations in the node 3. △

It is worth mentioning that there is always a finite number of distinct splits. For numerical and ordered data from learning sample $\mathcal{L}$ of size N, we have at most $N-1$ possibilities to split. We can find them simply by ordering the data and choosing midpoints between two consecutive distinct values of the desired variable.

For a categorical variable from learning sample $\mathcal{L}$ of size N, there are $2^{N-1}-1$ possible combinations to split N data into two indistinguishable sets. And as it grows exponentially, it quickly becomes computationally infeasible as for $N = 31$, there are over a billion possible combinations.

Now back to answering the question of how to select splits. At node t, the algorithm searches through the variables one by one, starting with X_1 and going to X_M. For each variable it finds the best split s and calculates the decrease in the impurity of the split $\Delta i(s,t)$. It then compares all M possible splits and selects the best one, i.e., the one with largest decrease in impurity $\Delta i(s,t)$.

1.3.2 What class to assign to a terminal node?

To discuss this topic, we first need a precise definition of what a class assignment is.

Definition 17 (Class assignment rule)
Let $\mathcal{T}$ be a binary tree-structured classifier with J classes. A function $q: \mathcal{T} \to \{1, 2, \ldots, J\}$ is called a class assignment rule *if it assigns a class $j \in \{1, 2, \ldots, J\}$ to a node $t \in \mathcal{T}$ and it will be denoted by $q(t)$.*

Similarly to the very beginning of the thesis, where we defined the true misclassification rate, we will define the probability of misclassification of a node.

Misclasification probability

Definition 18 (Probability of misclassification)
Let $\mathcal{T}$ be a binary tree-structured classifier with J classes, and q be a class assignment rule. The probability of misclassification *for node $t \in \mathcal{T}$ and class assignment rule q is given by*

$$r_q^*(t) = \sum_{j \neq q(t)} P(j \mid t).$$

The resubstitution estimate of the probability of misclassification is given by

$$r_q(t) = \sum_{j \neq q(t)} p(j \mid t).$$

It is easy to see that the estimate $r_q(t)$ is minimized for class assignment rule $q^*(t) = \operatorname{argmax}_{i \in \{1,2,\ldots\}} p(i \mid t),\ t \in \mathcal{T}$. If the maximum is achieved for two or more different classes $q^*(t)$ assigns class arbitrarily as any of the maximizing cases.

Thus we can omit the choice of class assignment rule because it will always be the same and write only

$$r(t) = 1 - \max_{j \in \{1,2,\ldots,J\}} p(j \mid t).$$

Recall from Definition 5 that the true misclassification rate of a classifier d is denoted by $R^*(d)$ and can be estimated by its resubstitution estimate $R(d) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(d(\mathbf{X}_i) \neq Y_i)$.

Our task now will be to find a similar estimate for the true misclassification rate of a binary tree-structured classifier $\mathcal{T}$. The calculation follows directly from the general probability rule in the following way:

$$R(\mathcal{T}) = \sum_{t \in \tilde{\mathcal{T}}} r(t)p(t),$$

i.e., by the sum over all terminal nodes of the product of the probability of misclassification given that the observation was classified to node t ($r(t)$) and the probability of a case being classified to node t ($p(t)$).

We will now denote for $t \in \mathcal{T}$: $R(t) = r(t)p(t)$. Then the resubstitution estimate of the overall misclassification rate of the binary tree-structured classifier $\mathcal{T}$-$R^*(\mathcal{T})$ is

$$R(\mathcal{T}) = \sum_{t \in \tilde{\mathcal{T}}} R(t).$$

1.3.3 How to stop splitting?

An important observation for this section will be formulated as the following theorem.

Theorem 3
Let $\mathcal{T}$ be a binary tree-structured classifier and let $\mathcal{T}'$ be a binary tree-structured classifier obtained from $\mathcal{T}$ by splitting any of the terminal nodes of $\mathcal{T}$, then $R(\mathcal{T}') \leq R(\mathcal{T})$.

This theorem comes from the formula $R(\mathcal{T}) = \sum_{t \in \tilde{\mathcal{T}}} R(t)$ and from the fact that for $t \in \mathcal{T}$ it holds $R(t) \geq R(t_L) + R(t_R)$. The proof of this can be found in Breiman et al. [1993] at page 95.

After Definition 15, we suggested that one way how to stop splitting is when the decrease in impurity does not exceed a certain threshold, i.e., if $\Delta i(s,t) < \beta$, for $\beta \geq 0$. However, this approach is not really useful if β is too low, there are too many splits, and the tree is too large. On the other hand, for increasing β, there may appear a situation when the best possible split s for node $t \in \mathcal{T}$ has a small decrease in impurity $\Delta i(s,t)$, so we stop splitting.

However, if we did perform this split, there would be much better splits s_L and s_R of the nodes t_l and t_R with large $\Delta i(s_L, t_L)$ and $\Delta i(s_R, t_R)$ coming after the split s that are not performed because we stopped splitting at node t.

It was discovered that looking for the right stopping rule is the wrong way of looking at the problem. A much more satisfactory approach is to grow a tree that is much too large (we will explain how to do that soon) and then *prune* (we will

again explain precisely what that means) it upwards in the right way until it is pruned back to the root node. We will define the terms precisely in this section.

The first step is to grow a very large tree, also called *initial tree,* that will be denoted by $\mathcal{T}_{max}$, by letting the splitting procedure continue until all terminal nodes are either small, pure or they consist of identical vectors of covariates.

The ideal case would be to grow the tree until each terminal node contains only 1 case with identical vectors of covariates. However, when it is not computationally feasible, it is possible to use a compromise method. This method also grows sufficiently large initial trees by specifying a number $N_{\min}$ and continuing to split until each terminal node $t \in \tilde{\mathcal{T}}$ is pure, satisfies the condition $N(t) \leq N_{\min}$, i.e., there are at most $N_{\min}$ observations in the node t, or it contains only identical vectors of covariates.

After growing the tree $\mathcal{T}_{\max}$, it then *prunes* the tree upwards, i.e., it produces a sequence of subtrees of $\mathcal{T}_{\max}$ and after a finite number of steps eventually collapses to the root node $\{1\}$ itself. We will define pruning now.

Definition 19 (Pruning)
Let $\mathcal{T}$ be a binary tree-structured classifier, $t \in \mathcal{T}$ and $\mathcal{T}_t$ a branch of $\mathcal{T}$. We will call deleting all descendants of node t from the tree $\mathcal{T}$ pruning a branch $\mathcal{T}_t$. *The tree pruned this way will be denoted by $\mathcal{T} - \mathcal{T}_t$.*

A binary tree-structured classifier $\mathcal{T}'$ that was obtained from $\mathcal{T}$ by successive pruning will be called a pruned subtree *of $\mathcal{T}$ and it will be denoted by $\mathcal{T}' < \mathcal{T}$.*

Naturally, pruning only makes sense if t is a non-terminal node.

Example. In our example, pruning can be illustrated by deleting the nodes 2 and 3 and thus having all observations in node 1. △

Even for relatively small sizes of $\mathcal{T}_{max}$, there are incredibly many different ways to prune up to the root node. Therefore it is necessary to find a way to selectively prune the tree so that each subtree selected is the *best* subtree in its size.

We need to specify what we mean by *best.* As one may think, we will again measure the goodness of a subtree by the resubstitution estimate $R(\mathcal{T})$.

The selective pruning starts with a given initial tree $\mathcal{T}_{\max}$. It completely disregards how the tree $\mathcal{T}_{\max}$ was grown, what splitting criterion was chosen etc. It computes $R(t)$ for each $t \in \mathcal{T}_{\max}$ and progressively prunes $\mathcal{T}_{\max}$ such that at each step of the pruning, $R(\mathcal{T})$ is as small as possible. We will denote the pruned subtree of tree $\mathcal{T}$ after j-th pruning by $\mathcal{T}_j$.

However, this approach has a severe disadvantage. The sequence of subtrees $\mathcal{T}_{\max}, \mathcal{T}_1, \mathcal{T}_2, \ldots$ is not descending, i.e., it does not necessarily hold that $\mathcal{T}_{n+1}$ is a subtree of $\mathcal{T}_n$. For this reason, a different approach was adopted, and we will devote the following section to this approach.

1.3.4 Minimal cost-complexity pruning

This approach will be called *minimal cost-complexity pruning*, and we will start by defining what *complexity* means and then define the minimal cost-complexity pruning.

Definition 20 (Complexity, cost-complexity)
Let $\mathcal{T}$ be a binary tree-structured classifier. We will call the number of its terminal nodes $|\tilde{\mathcal{T}}|$ the complexity of tree $\mathcal{T}$. *Also let $\alpha \geq 0$, that will be called* complexity parameter *and the quantity $R_\alpha(\mathcal{T}) = R(\mathcal{T}) + \alpha|\tilde{\mathcal{T}}|$ will be called* cost-complexity.

It is easy to see that $R_\alpha(\mathcal{T})$ is a linear combination of the cost of the tree $R(\mathcal{T})$ and its complexity $|\tilde{\mathcal{T}}|$.

Our task now will be to find for each $\alpha \in [0, +\infty)$ a pruned subtree $\mathcal{T}(\alpha) < \mathcal{T}_{\max}$ that minimizes R_α.

By a simple observation, we can see that for small values of α, the cost for having more terminal nodes is small, so the tree $\mathcal{T}(\alpha)$ will be large. As the complexity parameter α increases, the tree $\mathcal{T}(\alpha)$ minimizing R_α will have fewer and fewer terminal nodes. Obviously, for $\alpha \to +\infty$, $\mathcal{T}(\alpha)$ will consist only of the root node $\{1\}$.

Even though α is running through the non-negative real line $[0, +\infty)$, it is easy to see that the tree $\mathcal{T}_{\max}$ has only a finite number of terminal nodes, so there will be only a finite sequence subtrees $\mathcal{T}_{\max}, \mathcal{T}_1, \mathcal{T}_2, \ldots$. Because of the finiteness, what happens is that for a given α if $\mathcal{T}(\alpha)$ is the minimizing tree, then it continues to be the minimizing tree until we find $\alpha' > \alpha$ such that $\mathcal{T}(\alpha') < \mathcal{T}(\alpha)$ is now minimizing and continues to be minimizing until we find a different value $\alpha'' > \alpha'$ and so on.

The process of pruning is easy to describe, but we need to answer some important questions:

1. Is there a unique subtree $\mathcal{T} < \mathcal{T}_{max}$ minimizing $R_\alpha(\mathcal{T})$ for each value of $\alpha \in [0, +\infty)$?

2. Is the sequence of subtrees $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \ldots$ decreasing (in the subset sense)?

3. Is there a computationally feasible algorithm for the pruning process?

Let us answer the first question now. We will start by defining a *smallest minimizing subtree.*

Definition 21 (Smallest minimizing subtree)
Let $\mathcal{T}_{max}$ be an initial tree and let $\alpha \in [0, +\infty)$ be a complexity parameter. We say that $\mathcal{T}(\alpha)$ is the smallest minimizing subtree *of the initial tree tree $\mathcal{T}_{max}$ for given α, if both following conditions hold:*

- $R_\alpha(\mathcal{T}(\alpha)) = \min_{\{\mathcal{T}; \mathcal{T} < \mathcal{T}_{max}\}} R_\alpha(\mathcal{T})$
- *if $R_\alpha(\mathcal{T}) = R_\alpha(\mathcal{T}(\alpha))$, then $\mathcal{T}(\alpha) < \mathcal{T}$.*

The second condition states that if trees $\mathcal{T}$ and $\mathcal{T}(\alpha)$ have the same cost-complexity, then $\mathcal{T}(\alpha)$ must be a pruned subtree of $\mathcal{T}$, i.e., it breaks ties as it says that if two trees have the same cost complexity, then the pruned subtree is chosen.

Theorem 4
For every value of α, there exists a smallest minimizing subtree.

Proof. The proof can be found in Breiman et al. [1993] on the page 68. □

For the next section, we will need the following statement: $R(t) \geq R(t_L) + R(t_R)$. It intuitively holds because if $R(t) < R(t_L) + R(t_R)$ held, it would mean that we performed a split that increased the misclassification costs. The formal proof of this can be found in Breiman et al. [1993] on page 96.

To start pruning the tree, we need to do one more adjustment. We do not begin with pruning the tree $\mathcal{T}_{\max}$ but rather $\mathcal{T}_1 = \mathcal{T}(0)$ that is the smallest (in the sense of inclusion) subtree of $\mathcal{T}_{\max}$ satisfying that $R(\mathcal{T}_1) = R(\mathcal{T}_{\max})$.

$\mathcal{T}_1$ is obtained from $\mathcal{T}_{\max}$, by pruning branches $\mathcal{T}_t$ for such $t \in \mathcal{T}_{\max}$ such that t is the father node of terminal nodes t_L and t_R, that satisfy $R(t) = R(t_L) + R(t_R)$.

Thanks to this adjustment and the previous statement, it now holds $R(t) > R(\mathcal{T}_t)$ for each $t \in \mathcal{T} - \tilde{\mathcal{T}}$, i.e., for each nonterminal node.

1.3.5 Weakest-link cutting

For $\alpha \in [0, +\infty)$ and non-terminal node t of the tree $\mathcal{T}_1$ it holds

$$R_\alpha(\{t\}) = R(t) + \alpha.$$

From the previous statement, we also know that $R(t) > R(\mathcal{T}_t)$. It is then easy to see that there exists a value of α that we will denote by α_1 for which it holds:

- $R_\alpha(\{t\}) > R_\alpha(\mathcal{T}_t)$ for $\alpha < \alpha_1$
- $R_\alpha(\{t\}) = R_\alpha(\mathcal{T}_t)$ for $\alpha = \alpha_1$
- $R_\alpha(\{t\}) < R_\alpha(\mathcal{T}_t)$ for $\alpha > \alpha_1$.

(by $\{t\}$ we denote tree consisting only of its root node t) and

$$R_\alpha(\mathcal{T}_t) = R(\mathcal{T}_t) + \alpha|\tilde{\mathcal{T}}_t|.$$

Thus we prefer the branch $\mathcal{T}_t$ over the node t for $\alpha < \alpha_1$. However, if $\alpha \geq \alpha_1$, then we prefer the node t over the branch $\mathcal{T}_t$ (in case of equality, we prefer trees with fewer terminal nodes).

To find the value of α_1 we need to solve following inequality $R_\alpha(\mathcal{T}_t) < R_\alpha(\{t\})$ from which we get

$$\alpha < \frac{R(t) - R(\mathcal{T}_t)}{|\tilde{\mathcal{T}}_t| - 1}.$$

The statement $R(t) > R(\mathcal{T}_t)$ guarantees positivity.

So we will choose $\alpha_1 = \frac{R(t) - R(\mathcal{T}_t)}{|\tilde{\mathcal{T}}_t| - 1}$. This means we found the value of α such that we would be better off pruning some branches. Now we need to find which branch to prune. For this, we define a function: $g_1(t) : \mathcal{T} \to \mathbb{R}$ such that

$$g_1(t) = \begin{cases} \frac{R(t) - R(\mathcal{T}_t)}{|\tilde{\mathcal{T}}_t| - 1} & \text{for } t \notin \tilde{\mathcal{T}}_1 \\ +\infty & \text{for } t \in \tilde{\mathcal{T}}_1. \end{cases}$$

And we will prune node t_1 such that $g_1(t_1) = \min_{t \in \mathcal{T}_1} g_1(t)$. Now let us use tree $\mathcal{T}_2 = \mathcal{T}_1 - \mathcal{T}_{t_1}$. The node t_1 will be called *the weakest-link*, and it is meant in the sense that for the complexity parameter α increasing from 0, $\mathcal{T}_{t_1}$ will be the first

branch to be pruned off, because $R_\alpha(\{t\})$ becomes less than or equal to $R_\alpha(\mathcal{T}_t)$, so thee node t becomes preferable to the branch $\mathcal{T}_t$.

When we apply this approach recursively, i.e., we define the function

$$g_i(t) = \begin{cases} \frac{R(t)-R(\mathcal{T}_t)}{|\mathcal{T}_t|-1} \text{ for } t \notin \tilde{\mathcal{T}}_i \\ +\infty \text{ for } t \in \tilde{\mathcal{T}}_i. \end{cases}$$

and thanks to the function, find the i-th weakest-link t_i such that $g_i(t_i) = \min_{t \in \mathcal{T}_i} g_i(t)$ and the value of α: $\alpha_i = g_i(t_i)$ and finally prune it of by stating $\mathcal{T}_{i+1} = \mathcal{T}_i - \mathcal{T}_{t_i}$ for $i \in \mathbb{N}$. And in case there happens to be a tie such that $g_k(t_k) = g_k(t'_k)$, then we prune off both branches, i.e., we take $\mathcal{T}_{k+1} = (\mathcal{T}_k - \mathcal{T}_{t_k}) - \mathcal{T}_{t'_k}$.

The last contribution in this section will be the following statement:

Let $\{\alpha_k\}_{k=0}$ be the sequence of α's found by the algorithm we just described. Then it holds $\alpha_0 = 0$ and for $k \in \mathbb{N}$ and $\alpha \in [\alpha_k, \alpha_{k+1})$: $\mathcal{T}(\alpha_k) = \mathcal{T}_k = \mathcal{T}(\alpha)$.

This statement describes how minimal cost-complexity pruning works. It starts with the tree $\mathcal{T}_1$, finds the weakest-link branch $\mathcal{T}_{t_1}$, and prunes it to get $\mathcal{T}_2$ when α reaches α_1 and then recursively continues with the calculations.

This approach also fulfills the last criterion, i.e., computational feasibility. It is not hard to calculate because the value of the function $g_i(t)$ does not depend on i as long as the node t is a part of the tree $\mathcal{T}_i$, which has to hold as it was not pruned yet. Thus we only need to calculate the value of $g_i(t)$ once for $i \in \mathbb{N}$.

The parameter $\alpha \in [0, +\infty)$ is then chosen by K-fold cross-validation in the following way:

- Randomly divide the training sample $\mathcal{L}$ into K subsets of similar sizes (sizes of different subsets differ at most by one). Let us denote these subsets by $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_K$.

  Now for $k \in \{1, 2, \ldots, K\}$, the method uses all subsets apart from the k-th one, that is $\mathcal{L} \setminus \mathcal{L}_k$ as a learning sample.

- Based on this learning sample, build a large tree $\mathcal{T}_{\max}$.

- Apply cost-complexity pruning with cost parameter α to $\mathcal{T}_{\max}$ to obtain $\mathcal{T}(\alpha)$ for $\alpha \in [0, +\infty)$.

- Evaluate the prediction error with the chosen impurity measure, e.g., Gini impurity or cross-entropy.

- Average the results for each value of α, and pick the value of α so that it minimizes the average error calculated by the Gini impurity or cross-entropy.

A leave-one-out cross-validation technique would be so computationally exhausting here that 10-fold cross-validation is usually used instead.

On the use of `cp` in `rpart`

This subsection is based on Greenwell [2022]. In the statistical software `R` and its implementation `rpart`, it is more common to use complexity parameter `cp` rather than α.

$$R_{cp}(\mathcal{T}) = R(\mathcal{T}) + cp \times |\tilde{\mathcal{T}}| \times R(\mathcal{T}_1)$$

The main advantage of this approach is that the scale now does not matter. The choice of $cp = 0$ results in the full tree $\mathcal{T}_0$, and the choice of $cp = 1$ results in a tree with zero splits, i.e., the root node 1.

1.4 Regression trees

So far, we have discussed classification problems, i.e., problems where the response variable Y is nominal or categorical. Our task now will be to build up theory also for problems where the response variable Y is continuous.

Fortunately, most of the theory from classification trees stays the same also for regression trees. We will now define what is different. However, we will use the same notation so that we can jump back and forth from regression to classification trees. Thus the reader needs to keep track of what kind of problem we are solving at the moment.

We will start with the feature space and response variables as we have to define continuous response variables.

Definition 22 (Feature space, vector of covariates and response variables)
Let $\mathbf{X} = (X_1, X_2, \ldots, X_M)^\top$ *be a random vector with values in a general* M*-dimensional space* $\mathcal{X} = \mathcal{X}^1 \times \mathcal{X}^2 \times \ldots \times \mathcal{X}^M$*. We will call* $\mathbf{X}$ *a* feature vector *or* vector of covariates *and* $\mathcal{X}$ *a* feature space.

Moreover, let Y *be a real random variable. It will be called* response variable, *and the pair* $(\mathbf{X}, Y)^\top$ *will be called feature-response pair.*

The probability space stays the same as in classification trees with the minor adjustment that the set $\{1, 2, \ldots, J\}$ *is now replaced by* $\mathbb{R}$*, and all terms related to it change as well.*

Now we can move on to redefining the term *classifier.* As the name suggests, classifiers are used for classification. However, our goal is not to classify a nominal or categorical random variable but to predict a continuous random variable. Thus we will call the function d *predictor.*

Definition 23 (Predictor)
Let $\mathcal{X}$ *be a feature space and* Y *be a response variable. A function* $d\colon \mathcal{X} \to \mathbb{R}$ *is called a* predictor *if the mapping* $d\colon \mathbf{X} \mapsto d(\mathbf{X})$ *assigns for each* $\mathbf{X} \in \mathcal{X}$ *some* $y \in \mathbb{R}$.

The definition of the learning sample is almost the same as in section 1.2. We only have to take into account that the response variable is continuous.

1.4.1 Goodness of a predictor

Now the most obvious difference between classification and regression trees comes. In classification problems, the goodness of a classifier was measured by the actual misclassification rate (Definition 5), i.e., as the probability of misclassifying. However, because the response variable is now continuous, we have to use a different way to measure the goodness of a predictor. We will use *mean squared error* for this purpose.

Definition 24 (Mean squared error)
Let $\mathcal{L}$ be a learning sample, and d be a predictor on the feature space $\mathcal{X}$. Also, let $(\mathbf{X}, Y)^\top$ be a feature-response pair, where $\mathbf{X}$ is from $\mathcal{X}$ and Y is a response variable, drawn randomly from the same probability distribution.

$$P(A, y) = \mathbb{P}(\mathbf{X} \in A, Y = y), A \subseteq \mathcal{X}, y \in \mathbb{R}$$

independent of $\mathcal{L}$. The mean squared error of predictor d is defined as

$$R^*(d) = \mathbb{E}_{\mathbb{P}}[(Y - d(\mathbf{X}))^2].$$

During the evaluation of the expectation $\mathbb{E}_{\mathbb{P}}[(d(\mathbf{X}) - Y)^2]$, we consider the learning sample $\mathcal{L}$ to be fixed. Thus a more precise notation would be

$$\mathbb{E}_{\mathbb{P}}[(d(\mathbf{X}) - Y)^2 \mid \mathcal{L}].$$

The mean squared error is a theoretical quantity that needs to be somehow estimated. We will again incorporate three methods of estimation similar to classification trees.

The resubstitution estimate

Let $\mathcal{L} = \left\{(\mathbf{X}^1, Y^1)^\top, (\mathbf{X}^2, Y^2)^\top, \ldots, \left(\mathbf{X}^N, Y^N\right)^\top\right\}$ be the learning sample of size N and d be a predictor built based on the learning sample $\mathcal{L}$..

The resubstitution estimate of the predictor d based on the learning sample $\mathcal{L}$ will be denoted by $R(d)$ and calculated using the following formula.

$$R(d) = \frac{1}{N} \sum_{i=1}^{N} (Y_i - d(\mathbf{X}_i))^2.$$

Again, the drawback of the resubstitution estimate is that it uses the same data as it used for constructing the predictor d instead of an independent sample. Thus it could give an overly optimistic estimation of the accuracy of d.

Similar to the classification problem, we also define test sample estimation and cross-validation estimation for regression problems.

Test sample estimation

This method is based on randomly dividing the learning sample $\mathcal{L}$ into two sets: $\mathcal{L}_1$ and $\mathcal{L}_2$ of sizes N_1 and N_2, where $\mathcal{L} = \mathcal{L}_1 \bigcup \mathcal{L}_2$ and $\mathcal{L}_1 \bigcap \mathcal{L}_2 = \emptyset$ (and thus $N_1 + N_2 = N$). Let d be a binary tree-strucutred predictor built based on the learning sample $\mathcal{L}$.

We use only the sample $\mathcal{L}_1$ to construct the predictor d and then we use the sample $\mathcal{L}_2$ to estimate $R^*(d)$. The test sample estimate (denoted by $R^{ts}(d)$) is calculated via:

$$R^{ts}(d) = \frac{1}{N_2} \sum_{\{i:(x_i,y_i)\in\mathcal{L}_2\}} (Y_i - d(\mathbf{X}_i))^2.$$

This again eliminates the drawback of the resubstitution method as it uses different data for constructing the estimate and for calculating its accuracy, but there arises another problem as we only use a part of the data for the construction

of d while the rest of the data had to be left out while constructing d so that it can be used independently in testing. This problem may be neglected when large datasets are used. In these cases, it is also preferred as it is much less computationally demanding than the last method.

Cross validation

This method randomly divides all data in $\mathcal{L}$ into K subsets of similar sizes (sizes of different subsets differ at most by one). Let us denote these subsets by $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_K$.

Now for $k \in \{1, 2, \ldots, K\}$, the method uses all subsets apart from the k-th one, that is $\mathcal{L} \setminus \mathcal{L}_k$ as a learning sample (this was denoted by $\mathcal{L}_1$ in the previous method) and $\mathcal{L}_k$ as $\mathcal{L}_2$ from the previous method. It calculates test sample estimate of k-th group denoted by $R^{ts}(d^{(k)})$.

Then it does the same for all $k \in \{1, 2, \ldots, K\}$ and calculates the cross-validation estimate $R^{CV}(d)$ as

$$R^{CV}(d) = \frac{1}{K} \sum_{k=1}^{K} R^{ts}(d^{(k)}),$$

i.e., it takes the average over all $k \in \{1, 2, \ldots, K\}$. The most popular modifications of cross-validations are, again, leave-one-out cross-validation and 10-fold cross-validation. The advantages and drawbacks of each method are the same as for classification trees.

Similarly to Definition 8 we will define *binary tree-structured predictors.*

Definition 25 (Binary tree-structured predictor)
A binary tree-structured predictor d *is a predictor constructed by repeatedly partitioning subsets of the feature space* $\mathcal{X}$ *(that will be denoted* $\mathcal{X}_1$*) into two descendant subsets.*

If we partition part of the feature space $\mathcal{X}$ *numbered* $k \in \mathbb{N}$*, then the two descendant subsets will be numbered* $2k$ *and* $2k + 1$*. Thus, we can write that* $\mathcal{X}_{2k} \bigcup \mathcal{X}_{2k+1} = \mathcal{X}_k$ *and* $\mathcal{X}_{2k} \bigcap \mathcal{X}_{2k+1} = \emptyset$*.*

Those subsets which are not partitioned anymore are called terminal subsets*. The terminal subsets form a partitioning of the feature space* $\mathcal{X}$*. To each vector of covariates* $\mathbf{X}$ *from a subset* $\mathcal{X}_t, t \in \mathbb{N}$ *is then predicted a value* $y(t)$ *based on all values in the subset* $\mathcal{X}_t$*.*

The partitions are formed by conditions on the coordinates of the covariates vector $\mathbf{X}$*.*

The binary tree-structured predictor predicts a class for each vector of covariates $\mathbf{X}$ *in the following way: We look at the questions laid upon the vector of covariates* $\mathbf{X}$ *and depending on the outcome of the question we decide to which subset it belongs to. The questions have to be from the standardized set of questions* $\mathcal{Q}$*. When the vector of covariates reaches the terminal subset* X_t*, it predicts value* $y(t)$*.*

By size of a region $\mathcal{X}_t$*, we will denote the number of observations from the training sample* $\mathcal{L}$ *contained in* $\mathcal{X}_t$*.*

Now we can finally switch to tree terminology the same way as in Definition 9. Also, Definition 10 stays the same, with the only change that $\mathcal{T}$ is now a binary tree-structured predictor instead of the classifier.

Lastly, we can come to the task of constructing a regression tree. This process is fairly similar to classification trees.

Construction of a regression tree

The construction of a regression tree starts, of course, with a learning sample $\mathcal{L}$. And similarly to classification trees, we need to determine the following:

- How to select a split?
- How to determine a node terminal?
- How to assign value to every terminal node?

1.4.2 How to assign a value to a node?

Let $\mathcal{L}$ be a learning sample. We need to find the predicted value for each terminal node t of tree $\mathcal{T}$ called $y(t)$ that minimizes the term

$$R(d) = \frac{1}{N}\sum_{i=1}^{N}(Y_i - d(\mathbf{X}_i))^2 = \frac{1}{N}\sum_{i=1}^{N}(Y_i - y(t))^2,$$

where t is a terminal node and $y(t)$ is predicted value in node t. That means we can split the observations according to the terminal nodes they belong to, i.e., the resubstitution estimate is

$$R(d) = \frac{1}{N}\sum_{t\in\tilde{\mathcal{T}}}\sum_{\{i;\mathbf{X}_i\in t\}}(Y_i - y(t))^2,$$

If we now have a look at each of the terminal nodes at once, we are trying to minimize the term

$$\sum_{\{i;\mathbf{X}_i\in t\}}(Y_i - y(t))^2.$$

Let $\bar{y}(t)$ denote

$$\bar{y}(t) = \frac{1}{N(t)}\sum_{i=1}^{N(t)} Y_i,$$

where $N(t)$ denotes the size of node t, i.e, the number of observations in that node. From the following calculation, we can see that

$$\begin{aligned}\sum_{\{i;\mathbf{X}_i\in t\}}(Y_i - y(t))^2 &= \sum_{\{i;\mathbf{X}_i\in t\}}(Y_i - \bar{y}(t) + \bar{y}(t) - y(t))^2 \\ &= \sum_{\{i;\mathbf{X}_i\in t\}}(Y_i - \bar{y}(t))^2 + \sum_{\{i;\mathbf{X}_i\in t\}}(\bar{y}(t) - y(t))^2 + 2(\bar{y}(t) - y(t))\sum_{\{i;\mathbf{X}_i\in t\}}(Y_i - (\bar{y}(t)).\end{aligned}$$

The last sum sums up to 0, and it is easy to see that the first and second sums are nonnegative and thus

$$\sum_{\{i;\mathbf{X}_i\in t\}}(Y_i - y(t))^2 = \sum_{\{i;\mathbf{X}_i\in t\}}(Y_i - \bar{y}(t))^2 + \sum_{\{i;\mathbf{X}_i\in t\}}(\bar{y}(t) - y(t))^2,$$

from which we get that

$$\sum_{\{i;\mathbf{X}_i\in t\}} (Y_i - y(t))^2 \geq \sum_{\{i;\mathbf{X}_i\in t\}} (Y_i - \bar{y}(t))^2,$$

and equality occurs if and only if

$$\bar{y}(t) = y(t).$$

To conclude, we will always assign the average of the response variables Y_i's to a given node. Now we can come up with the question of how to select a split.

1.4.3 How to select a split?

Similarly to classification trees, regression trees also have the task of splitting a node t into two son nodes t_L and t_R so that the son nodes are more *homogeneous* than the father node. However, instead of impurity (Definition 12) we are now trying to minimize the mean squared error or, particularly its resubstitution estimate $R(d)$.

We know that for the predictor d (also known as tree $\mathcal{T}$) it holds that:

$$R(\mathcal{T}) = \frac{1}{N} \sum_{t\in\tilde{\mathcal{T}}} \sum_{\{i;\mathbf{X}_i\in t\}} (Y_i - y(t))^2.$$

Thus for each node $t \in \mathcal{T}$ it holds that the resubstitution estimate of the mean squared error of node t is

$$R(t) = \frac{1}{N} \sum_{\{i;\mathbf{X}_i\in t\}} (Y_i - y(t))^2.$$

So we can conclude that the resubstitution estimate of the mean squared error of tree $\mathcal{T}$ is the sum of resubstitution estimates of mean squared errors of all terminal nodes $t \in \tilde{\mathcal{T}}$.

$$R(\mathcal{T}) = \sum_{t\in\tilde{\mathcal{T}}} R(t).$$

Now let us have a look at splits again. Similarly to classification trees, the goodness of a split will be quantified by the decrease in $R(\mathcal{T})$. If we look at one particular terminal node t, we can calculate its resubstitution estimate of the mean squared error $R(t)$. Let us assume that split s had split node t into two son nodes t_L and t_R. Then, the decrease of the resubstitution estimate of the mean squared error is

$$\Delta R(s,t) = R(t) - (R(t_l) + R(t_R)).$$

Let $\mathcal{T}'$ be the tree obtained from tree $\mathcal{T}$ by splitting node t by the split s into nodes t_L and t_R as mentioned before. Then it holds that

$$R(\mathcal{T}) = \sum_{u\in\tilde{\mathcal{T}}\setminus\{t\}} R(u) + R(t),$$

and

$$R(\mathcal{T}') = \sum_{u\in\tilde{\mathcal{T}}\setminus\{t\}} R(u) + R(t_L) + R(t_R).$$

Thus it holds that

$$R(\mathcal{T}) = R(\mathcal{T}') + \Delta R(s,t).$$

Now, similarly to classification trees, we will always select the split that gives the highest decrease of the resubstitution estimate $R(\mathcal{T})$, i.e., the split that best separates high response values from the low response ones.

When to stop splitting?

Fortunately, as we mentioned before, the pruning criterion does not depend on the way the tree was built. Thanks to that, we can adopt techniques described in the section 1.3.3 such as minimal cost-complexity pruning and weakest-link cutting.

This concludes the part of the thesis that extends the theory also to continuous response variable Y.

We will now briefly introduce bootstrap aggregating alias bagging.

1.5 Bootstrap aggregating

This section is based on Breiman [1996]. Bootstrap aggregating, or *bagging* in short, is an algorithm based on aggregating from bootstrapped samples.

It uses learning sample $\mathcal{L}$ of size N defined in Definition 4. In classification and regression trees, we used the learning sample to form a tree called $\mathcal{T}$ that was later used for predicting.

We will now make some adjustments to this technique. We will use a bootstrapped learning sample, i.e., we draw N cases from $\mathcal{L}$ with replacement to build a learning sample $\mathcal{L}^b$.

Then we build a classification or regression tree called $\mathcal{T}_b$ based on this learning sample $\mathcal{L}^b$.

Lastly, we repeat this for $B \in \mathbb{N}$ times with independently drawn bootstrapped samples $\mathcal{L}^1, \mathcal{L}^2, \ldots, \mathcal{L}^B$. As a result we get an ensamble of trees $\{\mathcal{T}_b\}_{b=1}^B$. From which we build our prediction.

For regression trees the prediction for a vector of covariates $\mathbf{X}$ will be the average, i.e.,

$$\frac{1}{B}\sum_{b=1}^{B} \mathcal{T}_b(\mathbf{X}).$$

For classification trees, the prediction for a vector of covariates $\mathbf{X}$ will be the plurality vote, i.e., the most common class $j \in \{1, 2, \ldots, J\}$ among the predictions $\{\mathcal{T}_b\}_{b=1}^B$. Again, in case of a tie, we arbitrarily chose one of the most common classes.

The main advantage of bootstrap aggregating over classification and regression trees is that it lowers the variance because it uses averages (or plurality votes) over more bootstrapped samples.

2. Regression trees in individual claims reserving

2.1 Individual claims reserving introduction

This section is based on Wüthrich and Merz [2015]. We will describe non-life individual claims reserving terms in greater detail.

One of the essential properties of non-life insurance is that the claims usually cannot be settled immediately. There are some delays in reporting the claims and also some delays in settling the claims. Because of these delays, we need to predict future cash flows of claims that have occurred in the past but are to be settled in the future. Based on the predictions, we price the future contracts. This prediction task is called *claims reserving*, and it assesses the *outstanding loss liabilities* of past claims.

Typically the most significant portion of the liability side of the balance sheet of a non-life insurance company is made of these claim reserves. Therefore it is very important to have a reasonable prediction of said reserves.

2.1.1 Claims reserving terms

We will introduce some basics of insurance claims terminology such as period insured, accident date, reporting delay, reporting date, settlement delay, and settlement date.

Let U_1, U_2 be times such that $U_2 > U_1$. U_1 is called the *beginning of insurance period* and U_2 is called the *end of insurance period.* We will always assume that $U_2 < +\infty$. The interval $[U_1, U_2]$ is called *period insured.* This period is specified in the insurance contract.

For our purposes, we will always assume that the beginning of the insurance period is 1, i.e., $U_1 = 1$ and the end of the insurance period will be denoted by $I \in \mathbb{N}$.

A non-life insurance claim is triggered by accident causing damage covered by an insurance contract. The date of the claim's occurrence is called *accident date*, and we will denote it by T_1. The insurance company is liable only for those claims whose accident date T_1 falls into the period insured $[U_1, U_2]$. For that reason, we are only interested in cases when the accident date falls into the period insured. We will thus neglect the cases when $T_1 \notin [U_1, U_2]$.

After the claim occurs, it has to be reported to the insurance company, and it is usually not reported immediately. It may take days or even years, depending on the nature of the insurance contract. As *reporting date* (denoted by T_2 and it holds $T_2 \geq T_1$), we will consider the date when all the necessary information is fully incorporated into the insurance company systems and thus is available for statistical analysis. The time elapsed between the claim's occurrence and its reporting to the insurance company $(T_2 - T_1)$ will be called *reporting delay.* It is worth mentioning that reporting date may be outside of the insured period, i.e., $T_2 > U_2$. Nevertheless, the company is liable for all claims with the accident date in the period insured.

When the claim is reported, it typically cannot be settled immediately, as the insurance company needs more detailed information about the claim. There can be several reasons for this delay. The insurance company may start an investigation for the details of the accident or wait for external information and bills from external companies to come.

Of course, the client would not be pleased if he had to wait several years for claim payments. For this reason, the insurance company pays out once some claims benefits are fully justified. These claim benefits payments form a cash flow sequence that takes place after the reporting date T_2. Our task will be to model these payments.

The time when the final assessment of a claim takes place is called *settlement time* or *closing time* and is denoted by T_3. It holds $T_2 \leq T_3$ and the interval $[T_2, T_3]$ is called *settlement period.* The difference $T_3 - T_2$ is called *settlement delay.* It is also possible for the claims to reopen after being settled.

If we denote today's time by t, we could have five different situations:

1. $t < U_1$. No insurance contract exists. There is nothing to be calculated;

2. $U_1 \leq t < T_1$

   There is an insurance contract, but no claims have yet occurred. The insurance company may be fortunate enough so that $T_1 > U_2$, i.e., the accident happens after the insurance period. That means that the insurance company is not liable unless renewal happens. The only information available at this time is that the insurance contract had been signed;

3. $T_1 \leq t < T_2$ and $U_1 \leq T_1 \leq U_2$

   The accident occurred during the insured period but has not been reported to the insurance company yet. These claims are called *Incurred But Not yet Reported* (IBNR) claims. The insurance company does not have any claim-specific data for claims from this group. It has only external information. It can be split into global data such as economic data (market situation) or worldwide relevant facts such as the COVID pandemic, war, nuclear power plant accident, and local data such as weather conditions (storms, floods, hailstorms, earthquakes) and economic (unemployment rate, interest rate and inflation of a country or region the insurance company is interested in). This external data could give a hint to the insurance company if they should expect more or fewer claims to be reported;

4. $T_2 \leq t < T_3$ and $U_1 \leq T_1 \leq U_2$

   The claims are reported to the insurance company, but the final assessment is missing. The insurance company is gathering data about individual claims. The more data it gathers, the more precise the prediction of the final assessment is. However, these claims are not completely settled yet. They are called *Reported But Not Settled* (RBNS) claims. The settlement period $[T_2, T_3]$ is the period within which cash flows are done.

   During the settlement period, the insurance company gathers more and more claim-specific information such as accident date, cause of the accident, type of the accident, location of the accident, line-of-business and insurance

contracts involved, claims assessments, medical assessments, already paid cash flows and so on in addition to the external information it had before;

5. $T_3 \leq t$ and $U_1 \leq T_1 \leq U_2$

   The claim is settled, the file is closed and stored, and we expect no further information or payment for this claim. In some cases, it may be necessary to reopen a closed file due to unexpected development.

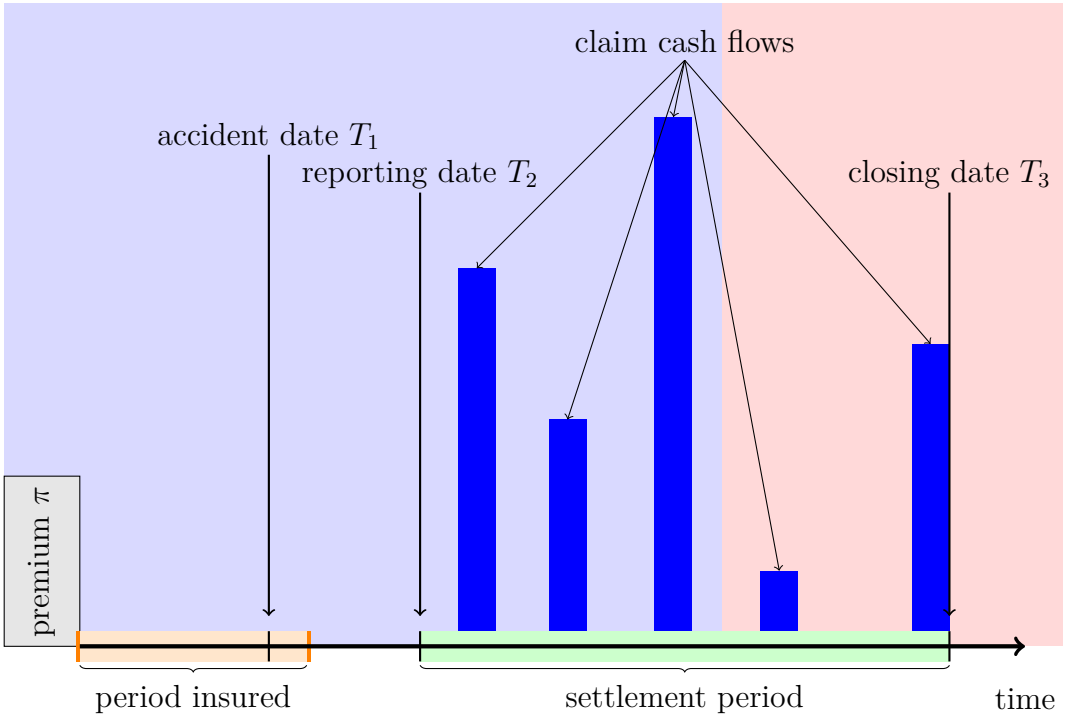


Figure 2.1: A non-life insurance claim progress illustration.

2.1.2 Notation

In order to predict and use statistical inference about insurance contracts and claims, we need to build as homogeneous groups as possible so that we can apply statistical tools such as the law of large numbers.

Once the homogeneous groups are built, we can study all claims that belong to the same group. These claims are further categorized by their accident date. Claims occurring in the same accident period may be triggered by similar external factors such as weather or economic situations.

Since the usual time scale for insurance contracts is years, the claims are typically gathered on a yearly time scale. Thus we consider calendar years denoted by $i \in \{1, 2, \ldots, I\}$. In our case, we will consider accounting years to be the same as calendar years, and thus we will refer to calendar years only. All claims with accident date $T_1 \in [1/1/i, 31/12/i]$ are called *claims with accident year i.* These claims generate cash flows which are also considered on a yearly level, i.e., all payments done in the same calendar year are aggregated.

We will start with the following notation. For the insurance period $[1, I]$, we will denote by

$$X$$

the sum of all payments done for claims in the insurance period.

We can divide these payments by their accident years. For fixed $i \in \{1, 2, \ldots, I\}$ let

$$X_i$$

denote the sum of all payments done for claims with accident year i. It holds

$$X = \sum_{i=1}^{I} X_i.$$

Naturally, we can split this quantity according to the reporting delay d, which means that for fixed $i \in \{1, 2, \ldots, I\}$ and $d \in \mathbb{N}_0$, we define

$$X_{i,d}$$

as the sum of all payments done for claims with accident year i reported in calendar year $i + d$. Then it holds

$$X_i = \sum_{d=0}^{+\infty} X_{i,d}.$$

We will also introduce a maximal reporting delay of $D \in \mathbb{N}_0$. This means that for all claims with reporting delay $d > D$ the insurance company is no more liable and thus $X_{i,d} = 0$. This results in the modification of the previous sum

$$X_i = \sum_{d=0}^{D} X_{i,d}.$$

Let $N_{i,d}$ denote the number of claims with accident year $i \in \mathbb{N}$ and reporting delay of $d \in \mathbb{N}_0$, i.e., the number of claims that occurred in calendar year i and were reported in calendar year $i + d$.

Then, for $v \in \{1, 2, \ldots, N_{i,d}\}$, we will define the sum of all payments for claim v-th claim of accident year i and reporting delay d by:

$$X_{i,d}^{(v)}$$

and it holds that

$$X_{i,d} = \sum_{v=1}^{N_{i,d}} X_{i,d}^{(v)},$$

as we defined $X_{i,d}$ as the sum all payments with given accident year i and reporting delay d. Now we can address each claim by the triplet (i, d, v).

Furthermore, each of these claims generates a sequence of payments that will be denoted by

$$X_{i,d\,|\,0}^{(v)}, X_{i,d\,|\,1}^{(v)}, \ldots,$$

where $X_{i,d\,|\,k}^{(v)}$ denotes the payments done in calendar year $i + d + k$ for claim (i, d, v). Now we can address each claim payment by the quadruplet (i, d, k, v) and it holds

$$X_{i,d}^{(v)} = \sum_{k=0}^{+\infty} X_{i,d\,|\,k}^{(v)}.$$

The main task of the thesis will be to model these payments.

For that, we need a little bit different approach. We need to look at the claim payments, not as payments in year $i + d + k$ but rather as payments $l = d + k$ years after the accident year i.

We will also introduce *maximal settlement delay* L. That means that for claim payments (i, d, k, v) such that $d + k > L$, we again set $X^{(v)}_{i,d \mid k} = 0$.

Thus we can again simplify the previous sum

$$X^{(v)}_{i,d} = \sum_{k=0}^{L-d} X^{(v)}_{i,d \mid k}$$

Altogether, it holds that

$$X = \sum_{i=1}^{I} X_i = \sum_{i=1}^{I} \sum_{d=0}^{D} X_{i,d} = \sum_{i=1}^{I} \sum_{d=0}^{D} \sum_{v=1}^{N_{i,d}} X^{(v)}_{i,d} = \sum_{i=1}^{I} \sum_{d=0}^{D} \sum_{k=0}^{L-d} \sum_{v=1}^{N_{i,d}} X^{(v)}_{i,d \mid k}.$$

As we suggested before, the quantity X_i can also be expressed differently. Let

$${}_lX_i$$

for $i \in \mathbb{N}$ and $l \in \mathbb{N}_0$ denote the sum of all payments for claims with accident year i paid in year $i + l$. Then it holds

$${}_lX_i = \sum_{d=0}^{l} \sum_{v=1}^{N_{i,d}} X^{(v)}_{i,d \mid l-d}$$

and for the total sum of claim payments with accident year i it holds

$$X = \sum_{i=1}^{I} X_i = \sum_{i=1}^{I} \sum_{l=0}^{L} {}_lX_i = \sum_{i=1}^{I} \sum_{l=0}^{L} \sum_{d=0}^{l} \sum_{v=1}^{N_{i,d}} X^{(v)}_{i,d \mid l-d}.$$

Lastly, one more modification brings a slightly different approach is the following let for $l \in \mathbb{N}$

$${}_lX$$

denote the sum of all payments in the year l, then it holds

$${}_lX = \sum_{i=\max\{1,l-L\}}^{\min\{l,I\}} {}_{l-i}X_i.$$

The sum of all payments is then expressed via ${}_lX$ in the following way:

$$X = \sum_{l=1}^{I+L} {}_lX$$

and the sum of all payments is calculated via

$$X = \sum_{l=1}^{I+L} {}_lX = \sum_{l=1}^{I+L} \sum_{i=\max\{1,l-L\}}^{\min\{l,I\}} {}_{l-i}X_i = \sum_{l=1}^{I+L} \sum_{i=\max\{1,l-L\}}^{\min\{l,I\}} \sum_{d=0}^{l-i} \sum_{v=1}^{N_{i,d}} X^{(v)}_{i,d \mid l-i-d}.$$

2.2 Outstanding loss liabilities

At the time $t \in \mathbb{N}$, the insurance company is liable for all claims that have occurred in accident years $i \leq t$.

These claims are called *past exposure claims.* The claims can be divided into three groups: closed ($t \geq T_3$), RBNS ($T_2 \leq t \leq T_3$), and IBNR ($T_1 \leq t < T_2$). Let us now assume that $t > L$, i.e., at the current time, at least for the claims with accident year $i = 1$, the maximal settlement period is over, and thus we are not expecting any further payments for claims within this accident year. Let t and L be fixed, and let us choose a stochastic basis

$$\left(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t=0}^{I+L}, \mathbb{P}\right)$$

with discrete filtration, $(\mathcal{F}_t)_{t=0}^{I+L}$ that is an increasing (in the subset sense) sequence of σ-algebras

$$\{\emptyset, \Omega\} = \mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \ldots \subseteq \mathcal{F}_{I+L} = \mathcal{F}.$$

$\mathcal{F}_t$ represents the information available at time $t \in \{0, 1, \ldots, I + L\}$. We will always assume that ${}_lX_i$ is $\mathcal{F}_{l+i}$-measurable, i.e., observable at time $t = l + i$ and integrable with respect to $\mathbb{P}$ for $i \in \{1, 2, \ldots, I\}, l \in \{0, 1, \ldots, L\}$.

At time $t \in \{0, 1, \ldots, I + L\}$ we have observed claim payments

$$\mathcal{D}_t = \{{}_lX_i : i + l \leq t, i \in \{1, 2, \ldots, I\}, l \in \{0, 1, \ldots, L\}\},$$

i.e., the payments from claims with accident years i paid l years later when $i + l \leq t$, in other words, all the payments that have happened up until time t.

The claims that occurred in the past but are not closed yet will generate cash flows in future calendar years. We will denote these cash flows by

$$\mathcal{D}_t^c = \{{}_lX_i : i + l > t, i \in \{1, 2, \ldots, I\}, l \in \{0, 1, \ldots, L\}\}.$$

This will be called *outstanding loss liabilities at time t*. Predicting these quantities will be the main task of this thesis.

When an insurance company ceases to issue new insurance contracts, i.e., $t \geq I$, a so-called *run-off* situation described by $\mathcal{D}_t$ and $\mathcal{D}_t^c$ arises. This situation is illustrated in Table 2.1 where $\mathcal{D}_t$ is depicted in blue and $\mathcal{D}_t^c$ in red. The insurance company no longer issues the insurance, so they cannot expect any premium payments in the future. Thus the insurance company needs to build enough reserves so that it is able to fulfill the future cash flows of past exposure claims. The resulting reserves for outstanding loss liabilities are called *claims reserves*. The claims reserves should satisfy that all past information should be used when evaluating the claims reserves and the claims reserves should also be the best estimate for the outstanding loss liabilities.

That means our goal is to predict $\mathcal{D}_t^c$ based on (all) available information $\mathcal{F}_t \supseteq \sigma(\mathcal{D}_t)$ at time $t \geq I$. It is often the case that $\mathcal{F}_t$ and $\sigma(\mathcal{D}_t)$ are identical, i.e., we have no further information than the cash flows. Particularly, our goal is to define a stochastic model on a stochastic basis $\left(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t=0}^{I+L}, \mathbb{P}\right)$ that satisfies:

1. it includes past information $\mathcal{F}_t$ through a filtration $(\mathcal{F}_t)_{t=0}^{I+L}$.
2. it reflects past observations $\mathcal{D}_t$
3. it can predict future cash flows $\mathcal{D}_t^c$ of the outstanding loss liabilities

accident year i	development year l					
	0	1	...	l	...	L
1	${}_0X_1$	${}_1X_1$		...		${}_LX_1$
⋮	⋮	⋮		⋱		⋮
$I-L$	${}_0X_{I-L}$	${}_1X_{I-L}$		...		${}_LX_{I-L}$
⋮						
i	⋮	⋮		⋱		⋮
⋮						
$I-1$	${}_0X_{I-1}$	${}_1X_{I-1}$		...		${}_LX_{I-1}$
I	${}_0X_I$	${}_1X_I$		...		${}_LX_I$

Table 2.1: An illustration of a run-off situation.

2.3 Solvency II

This short section briefly introduces what Solvency II is and what it says about Solvency Capital Requirement. It is based on European Union [2015].

Solvency II is a regulatory framework in the European Union implemented in 2016 for insurance companies and other financial institutions operating within the EU. It aims to ensure that the companies maintain adequate capital to cover their risks and remain financially solvent. One of the key elements of Solvency II is the Solvency Capital Requirement (SCR).

The Solvency Capital Requirement (SCR) shall be the amount of own funds that an undertaking needs to hold so that the probability of its capital being less than the SCR as a result of the risks to which it is exposed does not exceed a pre-defined level over a one-year time horizon. This pre-defined level shall be set at a level of 99.5% probability of adequacy over a one-year period.

2.4 The problem and the model considered

This section is based on Wüthrich [2018]. Let $\mathcal{F}_t$ denote the information available at time $t \in \mathbb{N}_0$. Then the prediction of the total nominal payments for reported claims of accident year i at time t is given by

$$\sum_{d=0}^{\min\{t-i,D\}} \sum_{v=1}^{N_{i,d}} \left(\sum_{k=0}^{\min\{t-i-d,L-d\}} X_{i,d\,|\,k}^{(v)} + \mathbb{I}(L > t-i) \sum_{k=t-i-d+1}^{L-d} \mathbb{E}\left[X_{i,d\,|\,k}^{(v)} \mid \mathcal{F}_t \right] \right).$$

Reported claims can be split into closed and RBNS. For closed claims, there are no further payments coming, so the last summation is equal to 0, and this the formula reduces to

$$\sum_{d=0}^{\min\{t-i,D\}} \sum_{v=1}^{N_{i,d}} \sum_{k=0}^{\min\{t-i-d,L-d\}} X_{i,d\,|\,k}^{(v)}$$

as we do not have to predict any future payments if we know that they are not coming. For RBNS claims, the last term is typically positive, as we expect some future claim payments. If $L \leq t - i$, then we have just reached the maximal development delay and thus, the second summation again reduces to zero.

The nominal payments for IBNR claims of accident year i at time t are predicted by

$$\sum_{d=t-i+1}^{D} \mathbb{E}\left[\sum_{v=1}^{N_{i,d}} \sum_{k=0}^{L-d} X_{i,d\,|\,k}^{(v)} \mid \mathcal{F}_t\right].$$

We will now state the model assumptions.

Assumption 1
Let $\left(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t=0}^{I+L}, \mathbb{P}\right)$ *be a stochastic basis with sufficiently rich and discrete filtration* $(\mathcal{F}_t)_{t=0}^{I+L}$.

Also let all processes $(N_{i,d})_{i,d}$ *be* $(\mathcal{F}_t)_{t=0}^{I+L}$*-adapted for time* $t = i + d$ *and* $(Y_{i,d\,|\,k}^{(v)})_{i,d,k,v}$ *be* $(\mathcal{F}_t)_{t=0}^{I+L}$*-adapted for time* $t = i + d + k$. *Furthermore, let us assume the following:*

(a1) claim counts $(N_{i,d})_{i,d}$ *and claim payments* $(X_{i,d\,|\,k}^{(v)})_{i,d,k,v}$ *are independent;*

(a2) the random variables $(N_{i,d})_{i,d}$ *and* $(X_{i,d\,|\,k}^{(v)})_{i,d,k,v}$ *are independent for different accident years* $i \in \{1, 2, \ldots, I\}$;

(a3) the processes $(X_{i,d\,|\,k}^{(v)})_k$ *are independent for different reporting delays* $d \in \{1, 2, \ldots, D\}$ *and different claims* $v \in \{1, 2, \ldots, N_{i,d}\}$;

(a4) The conditional distribution of $X_{i,d\,|\,k+1}^{(v)}$ *given* $\mathcal{F}_{i+d+k}$ *is*

$$X_{i,d\,|\,k+1}^{(v)} \mid \mathcal{F}_{i+d+k} \sim p_{d+k}\left(\mathbf{x}_{i,d\,|\,k}^{(v)}\right),$$

where $\mathbf{x}_{i,d\,|\,k}^{(v)}$ *is a realization of* $\mathbf{X}_{i,d\,|\,k}^{(v)}$, *which is a* $\mathcal{F}_{i+d+k}$ *measurable vector of covariates of the claim* (i, d, k, v). *It is contained in the feature space* $\mathcal{X}$ *that will be described soon, and* $p_{d+k} : \mathcal{X} \to [0, +\infty)$ *is some distribution.*

Remark. We will explain why we assume this now.

- The independence assumption (a1) is necessary to receive compound distributions;
- The independence assumption (a2) guarantees that claims from different accident year i can be modeled independently;

- The independence assumption (a3) ensures that we can model payments from a single claim $\mathbf{X}_{i,d}^{(v)}$ independently for all claims $v \in \{1, 2, \ldots, N_{i,d}\}$. This assumption is conditionally on the regression function $p_{d+k} : \mathcal{X} \to [0, +\infty)$;
- The most important assumption for the claims modeling is the conditional distribution given by the formula in (a4). It says that for every reported claim up to time $t = i + d + k$ there exists an $\mathcal{F}_{i+d+k}$-measurable realization $\mathbf{x}_{i,d \mid k}^{(v)}$ of the vector of covariates $\mathbf{X}_{i,d \mid k}^{(v)} \in \mathcal{X}$ that determines the conditional distribution of $X_{i,d \mid k+1}^{(v)}$. The probability function $p_{d+k} : \mathcal{X} \to [0, +\infty)$ can have any form. We calibrate it with machine learning techniques (mainly regression trees and bagging).

$\triangle$

2.5 Feature space and regression tree

2.5.1 Feature space

Under the model assumptions 1, the feature space consists of S-dimensional vectors

$$\mathbf{X}_{i,d \mid k}^{(v)} = \left(X_{i,d \mid k}^{(v)}(s)\right)_{s \in \{1,2,\ldots,S\}}^{\top} = \left(X_{i,d \mid k}^{(v)}(1), X_{i,d \mid k}^{(v)}(2), \ldots X_{i,d \mid k}^{(v)}(S)\right)^{\top} \in \mathcal{X}.$$

Some components of the covariates vectors are categorical, e.g., claim type, accident weekdays, or open/close status k years after the accident year, while others components of the covariates vectors are continuous, e.g., age, reporting delay, or payment k years after the accident year. In Table 2.2, we present the structure of the vector of covariates. The vector of covariates $\mathbf{X}_{i,d \mid k}^{(v)}$ has a realization $\mathbf{x}_{i,d \mid k}^{(v)}$ that we will now abbreviate to $\mathbf{x} = (x(1), x(2), \ldots x(S))^{\top} \in \mathcal{X}$.

$x(1)$	:	type of the insurance claim (`type`)
$x(2)$	:	age of the person insured (`age`)
$x(3)$	:	weekday of the accident (`weekday`)
$x(4)$	:	reporting delay (`d`)
$x(5)$	:	payment at time i (only present if $d = 0$) (`(X0)`)
$x(6)$	:	payment at time $i + 1$ (only present if $d \leq 1$) (`(X1)`)
	$\vdots$	
$x(5 + d)$	:	payment at time $i + d$ (`Xd+0`)
$x(5 + d + 1)$	:	payment at time $i + d + 1$ (`Xd+1`)
	$\vdots$	
$x(5 + d + k)$	:	payment at time $i + d + k$ (`Xd+k`)
$x(5 + d + k + 1)$	:	empty
	$\vdots$	
$x(S)$	:	empty

Table 2.2: Description of feature space components.

Remark. Some remarks to Table 2.2 and the feature space.

- The components $x(1)$, $x(3)$ are categorical, and they take values in $\{1,2,3,4,5,6\}$ and {Mon, Tue, Wed, Thu, Fri, Sat, Sun} respectively.

  All the other components are continuous, $x(2)$ takes values in $[18,65]$, $x(4)$ takes values in $\mathbb{N}_0$ and $x(5), x(6), \ldots, x(S)$ take values in $\mathbb{R}$.

  Thus we can conclude that the feature space holds

  $$\mathcal{X} = \{1,2,3,4,5,6\} \times [18,65] \times \{\text{Mo, Tu, We, Th, Fr, Sa, Su}\} \times \mathbb{N}_0 \times \mathbb{R}^{S-4}.$$

- The components $x(5), x(6), \ldots, x(5+d+k)$ model the payment history up to time $i+d+k$. The initial zeros correspond to reporting delay d, i.e., there were no payments because the claim was not even reported. This is a part of the $\mathcal{F}_{i+d+k}$ measurable claim. The components $x(5+d+k+1), x(5+d+k+2), \ldots, x(S)$ are kept empty as they will be observed and filled in later in the claims development process and are not available at time $i+d+k$.

- The feature $\mathbf{X}^{(v)}_{i,d\,|\,k}$ does not include any accident year i specific information. Thanks to that, the regression function $\mathbf{x}^{(v)}_{i,d\,|\,k} \mapsto p_{d+k}(\mathbf{x}^{(v)}_{i,d\,|\,k})$ can by applied to any accident year.

- This setup allows us to make predictions recursively *one-period ahead*.

△

2.6 Regression trees, and bagging

2.6.1 Regression trees in `rpart`

We estimate the regression function $p_{d+k} : \mathcal{X} \to \mathbb{R}$ using classification and regression tree (CART) techniques to achieve $\hat{p}_{d+k}$. These techniques were introduced in chapter 1. At first, we build a large binary tree. In the second step, the tree size (specifically the complexity parameter *cp*) is determined by 10-fold cross-validation and pruned.

The standardized binary split tree growing algorithm is already implemented in `R`. It is described in Greenwell [2022], and we will use the `R` library and command `rpart` in the following way:

```
tree <-rpart(Xd+k+1∼ type + age + weekday + d + X0 + ... + Xd+k,
             data=data_t, method="anova",
             control=rpart.control(cp=0))
```

This command says that we regress the response $X^{(v)}_{i,d\,|\,k+1}$ from the feature vector $\mathbf{x}^{(v)}_{i,d\,|\,k}$. The data `data_t` used at time t is given by all observations that are $\mathcal{F}_t$ measurable, i.e., that hold $i+d+k \le t$. Method applied is regression trees, thus we input `'anova'`. The term `control` specifies that we want to build a very large tree (`cp=0`) that was called $\mathcal{T}_1$ in chapter 1.

Now we find the value of complexity parameter `cp` that yields the smallest cross-validation error denoted by `xerror` in the following way:

```
best <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]
```

Lastly, we prune the tree with the computed complexity parameter `cp`:

```
pruned_tree <- prune(tree, cp=best).
```

2.6.2 Bagging in `bagging`

We will use bagging techniques to improve further the regression function p_{d+k}. These techniques were introduced in section 1.5. In `R` it is done in the following way:

```
bag <- bagging(Xd+k+1∼ type + age + weekday + d + X0 + ... + Xd+k,
               data=data_t, nbagg = 10000,
               control=rpart.control(cp=0))
```

We use the same formula and data as in `rpart`.

2.7 Individual claims reserving analysis

2.7.1 Data description

Due to confidentiality reasons, we cannot use the data used in Wüthrich [2018]. Fortunately, the author invented a generator of individual claims for claims reserving studies. This generator is described in Wang and Wüthrich [2022]. Now we will explain what the data consists of.

The dataset consists of 51,338 closed, 7,596 RBNS, and 2,436 IBNR claims observed in years $i \in \{1, 2, \dots, 10\}$. We adjusted the data to have the structure described in Table 2.2.

The generator also generated claim payments for $t > 10$. Therefore, we can compare our methods with the (generated) reality.

From the context of the paper, it is not clear which type of business the data come from.

2.7.2 Data exploration

In Figure 2.2, we can see some basic marginal distributions. In Figure 2.2(a), there is the distribution of type. The type takes values from the set $\{1, 2, 3, 4, 5, 6\}$, and there is no further specification of what each type means. We can see that type 1 claims are the most frequent, with almost 20,000 claims, while type 2 and type 3 are the least common, with around 3,000 claims.

Figure 2.2(b) shows us the histogram of age. Apart from the youngest group, which is significantly more common, and the oldest, which is considerably less common, we cannot spot any abnormalities.

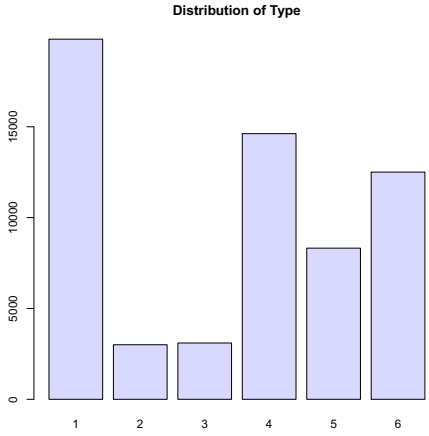
In Figure 2.2(c), we can see that accident years are slowly decaying to the end. However, it is not very significant.

Lastly, Figure 2.2(d) shows the accident weekdays are almost uniformly distributed. There are hardly any changes on weekdays.

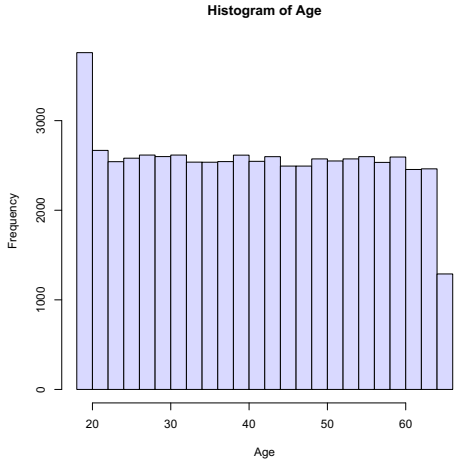
As mentioned before, for our data analysis, we have 58,934 reported claims of an insurance company over $I = 10$ accident years and maximal development delay of $L = 9$ years. That is, we have information available for claims (i, d, k, v) such that $i + d + k \leq I = 10$, which corresponds at time $t = I$ to

$$\mathcal{F}_t = \sigma\Big\{\mathbf{X}^{(v)}_{i,d\,|\,k}; i + d + k \leq t, v \leq N_{i,d}\Big\}.$$

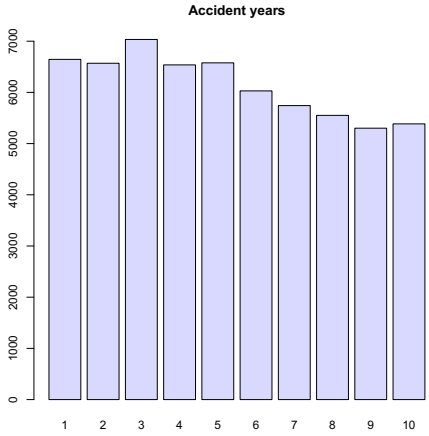
This gives us the formal definition of the filtration $(\mathcal{F}_t)_{t=0}^{I+L}$.



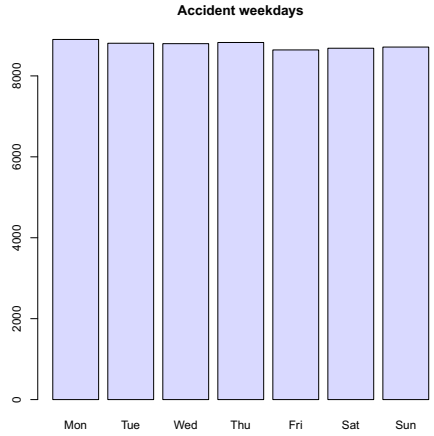
(a) Distribution of type among claims



(b) Distribution of the age of person injured



(c) Distribution of accident years



(d) Distribution of accident weekdays

Figure 2.2: Basic data exploration.

Our main goal is to predict

$$\mathcal{F}_t^c = \left\{ \mathbf{X}^{(v)}_{i,d \mid k}; i + d + k > t, v \leq N_{i,d} \right\}.$$

The last quantity can be split into two disjoint sets $\mathcal{F}_t^{RBNS}$ and $\mathcal{F}_t^{IBNR}$, where

$$\mathcal{F}_t^{RBNS} = \left\{ \mathbf{X}^{(v)}_{i,d \mid k}, i + d + k > t,\ v \leq N_{i,d},\ i + d \leq t \right\}$$

and

$$\mathcal{F}_t^{IBNR} = \left\{ \mathbf{X}^{(v)}_{i,d \mid k}, i + d + k > t,\ v \leq N_{i,d},\ i + d > t \right\}.$$

2.7.3 Model calibration for reported claims

Because any claim can be reopened at any time, we decided that we would not include any feature component that determines the openness of a claim. Thus,

all reported claims are considered to be RBNS. Nevertheless, this extension of modeling the openness/closeness of a claim is postponed to further research.

We will apply the regression tree growing algorithm and bagging algorithm to calibrate the probabilities p_l for the observed development periods $l \in \{0, 1, \ldots, L-1\}$, at time $t = I$, i.e., at the time the insurance company ceased to issue new insurance contracts. We set $l = d+k$, and thus $L = 9$ is the maximal observed development delay.

We consider for development periods $l = d + k$:

$$X^{(v)}_{i,d\,|\,k+1} \mid \mathcal{F}_{i+d+k} \sim p_{l=d+k}(\mathbf{x}^{(v)}_{i,d\,|\,k})$$

and for reported claims (i, d, k, v) such that $i + d + k + 1 \leq I = 10$. These are the claims that generate the filtration $\mathcal{F}_I$.

The calibration of p_l at time $t = I$ obtained in the previous section will now be used to predict the payments for reported claims.

Let us choose such a reported claim (i, d, v) such that $i + d \leq 10$, and corresponding payments (i, d, k, v) such that $i+d+k \leq 10$, so the vector of covariates $\mathbf{X}^{(v)}_{i,d\,|\,k}$ is observed at time $I = 10$. Now our goal is to predict

$$\left(X^{(v)}_{i,d,\,|\,k+1}, X^{(v)}_{i,d,\,|\,k+2}, \ldots\right).$$

Because we have also set the maximal development delay L to be 9, we can mainly focus on the prediction of

$$\left(X^{(v)}_{i,d,\,|\,k+1}, X^{(v)}_{i,d,\,|\,k+2}, \ldots, X^{(v)}_{i,d,\,|\,L-d}\right).$$

For $k' > k$ we predict $X^{(v)}_{i,d,\,|\,k'}$ at time $t = I = 10$ by

$$\hat{X}^{(v)}_{i,d\,|\,k'} = \mathbb{E}\left[X^{(v)}_{i,d,\,|\,k'} \mid \mathcal{F}_t\right]$$

And by the rule of total expectation, we get that

$$\mathbb{E}\left[X^{(v)}_{i,d,\,|\,k'} \mid \mathcal{F}_t\right] = \mathbb{E}\left[\mathbb{E}\ldots\left[\mathbb{E}\left[X^{(v)}_{i,d,\,|\,k'} \mid \mathcal{F}_{i+d+k'-1}\right] \mid \mathcal{F}_{i+d+k'-2}\right]\ldots \mid \mathcal{F}_I\right].$$

The last term is complicated to calculate analytically. Thus we use rather simulations.

If we replace the distribution p_l by its $\mathcal{F}_t$-measurable calibration $\hat{p}_l$ and consequently $\mathbb{E}$ by $\hat{\mathbb{E}}$, we can predict $X^{(v)}_{i,d,\,|\,k'}$.

This gives for each claim (i, d, v), such that $i + d \leq I$ at time $t = I = 10$ the estimate of the sum of all payments is

$$\hat{X}^{(v)}_{i,d} = \sum_{k=0}^{\min\{t-i-d,L-d\}} X^{(v)}_{i,d\,|\,k} + \mathbb{I}(L > t - i) \sum_{k=t-i-d+1}^{L-d} \hat{\mathbb{E}}\left[X^{(v)}_{i,d\,|\,k} \mid \mathcal{F}_t\right],$$

where $\hat{\mathbb{E}}$ denotes the expectation obtained from the calibration $\hat{p}_l$ at time I.

Similarly to earlier parts, we can express the values involved differently.

$${}_l\hat{X}^{RBNS}_i = \sum_{d=0}^{l} \sum_{v=1}^{N_{i,d}} \hat{X}^{(v)}_{i,d\,|\,l-d},$$

for $i \in \{1, 2, \ldots, I\},\ l \in \{1, 2, \ldots, L\},\ d \in \{0, 1, \ldots, D\}$, such that $i + l > I$ and $i + d \leq I$, where the first condition ensures that the payments are yet to come and the second condition ensures that the claims have been reported. These are the values for claims that have been reported from Table 2.1 depicted in red.

Then it for $l > I$ holds

$$ {}_l\hat{X}^{RBNS} = \sum_{i=\max\{1,l-L\}}^{\min\{l,I\}} {}_{l-i}\hat{X}_i^{RBNS}, $$

and

$$ \hat{X}^{RBNS} = \sum_{l=I+1}^{2I-1} {}_l\hat{X}^{RBNS}, $$

where $\hat{X}^{(v)}_{i,d\,|\,k}$ is a prediction for $X^{(v)}_{i,d\,|\,k}$ for claims (i, d, v) such that $i + d \leq I$ and payments (i, d, k, v) such that $i + d + k > I$.

	0	1	2	3	4	5	6	7	8	9
1	${}_0X_1$	${}_1X_1$	${}_2X_1$	${}_3X_1$	${}_4X_1$	${}_5X_1$	${}_6X_1$	${}_7X_1$	${}_8X_1$	${}_9X_1$
2	${}_0X_2$	${}_1X_2$	${}_2X_2$	${}_3X_2$	${}_4X_2$	${}_5X_2$	${}_6X_2$	${}_7X_2$	${}_8X_2$	${}_9\hat{X}_2$
3	${}_0X_3$	${}_1X_3$	${}_2X_3$	${}_3X_3$	${}_4X_3$	${}_5X_3$	${}_6X_3$	${}_7X_3$	${}_8\hat{X}_3$	${}_9\hat{X}_3$
4	${}_0X_4$	${}_1X_4$	${}_2X_4$	${}_3X_4$	${}_4X_4$	${}_5X_4$	${}_6X_4$	${}_7\hat{X}_4$	${}_8\hat{X}_4$	${}_9\hat{X}_4$
5	${}_0X_5$	${}_1X_5$	${}_2X_5$	${}_3X_5$	${}_4X_5$	${}_5X_5$	${}_6\hat{X}_5$	${}_7\hat{X}_5$	${}_8\hat{X}_5$	${}_9\hat{X}_5$
6	${}_0X_6$	${}_1X_6$	${}_2X_6$	${}_3X_6$	${}_4X_6$	${}_5\hat{X}_6$	${}_6\hat{X}_6$	${}_7\hat{X}_6$	${}_8\hat{X}_6$	${}_9\hat{X}_6$
7	${}_0X_7$	${}_1X_7$	${}_2X_7$	${}_3X_7$	${}_4\hat{X}_7$	${}_5\hat{X}_7$	${}_6\hat{X}_7$	${}_7\hat{X}_7$	${}_8\hat{X}_7$	${}_9\hat{X}_7$
8	${}_0X_8$	${}_1X_8$	${}_2X_8$	${}_3\hat{X}_8$	${}_4\hat{X}_8$	${}_5\hat{X}_8$	${}_6\hat{X}_8$	${}_7\hat{X}_8$	${}_8\hat{X}_8$	${}_9\hat{X}_8$
9	${}_0X_9$	${}_1X_9$	${}_2\hat{X}_9$	${}_3\hat{X}_9$	${}_4\hat{X}_9$	${}_5\hat{X}_9$	${}_6\hat{X}_9$	${}_7\hat{X}_9$	${}_8\hat{X}_9$	${}_9\hat{X}_9$
10	${}_0X_{10}$	${}_1\hat{X}_{10}$	${}_2\hat{X}_{10}$	${}_3\hat{X}_{10}$	${}_4\hat{X}_{10}$	${}_5\hat{X}_{10}$	${}_6\hat{X}_{10}$	${}_7\hat{X}_{10}$	${}_8\hat{X}_{10}$	${}_9\hat{X}_{10}$

Table 2.3: An illustration of how we learn to predict `X1`.

To conclude,

$$ {}_l\hat{X}_i^{RBNS} $$

represents the prediction of the sum of all payments (i, d, k, v), such that $d+k = l$ and $i + d + k = i + l > I$ for claims (i, d, v), such that $i + d \leq I$ with accident year i, being paid in year $i + l$.

The quantity

$$ {}_l\hat{X}^{RBNS} $$

represents the prediction of the sum of all payments (i, d, k, v), such that $d+k = l$, and $i + d + k = i + l > I$ for claims (i, d, v), such that $i + d \leq I$ being paid in year l.

Lastly the quantity

$$ \hat{X}^{RBNS} $$

represents the prediction of the sum of payments (i, d, k, v), such that $i+d+k > I$ for claims (i, d, v), such that $i + d \leq I$.

2.7.4 Prediction for reported claims

This means that for development period $l = d + k = 0$ we will use information $X^{(v)}_{i,0\,|\,0+1}$ and $\mathbf{X}^{(v)}_{i,0\,|\,0}$. That means we will regress `X1` as the response from variables `Type, Age, weekday, d, X0`. As it has to hold $i+d+k+1 \leq 10$ and $d = 0$ that means we can use data such that $i \leq 9$. This approach is illustrated in Table 2.3, where the known values of `X1` are in the lighter green while the corresponding values of `X0` are in darker green.

Now that our method has learned how to predict `X1`, we can apply this and predict ${}_1\hat{X}_{10}$. This is illustrated in Table 2.4, where we predict the values in light yellow based on the values in darker yellow.

For illustration, the optimal tree for $l = 0$ is depicted in Figure 2.3.

After the prediction of ${}_1\hat{X}_{10}$ we are in a situation depicted in Table 2.5, where we know the values in blue, have just predicted the value in yellow, and aim to predict the missing values in red.

	0	1	2	3	4	5	6	7	8	9
1	${}_0X_1$	${}_1X_1$	${}_2X_1$	${}_3X_1$	${}_4X_1$	${}_5X_1$	${}_6X_1$	${}_7X_1$	${}_8X_1$	${}_9X_1$
2	${}_0X_2$	${}_1X_2$	${}_2X_2$	${}_3X_2$	${}_4X_2$	${}_5X_2$	${}_6X_2$	${}_7X_2$	${}_8X_2$	${}_9\hat{X}_2$
3	${}_0X_3$	${}_1X_3$	${}_2X_3$	${}_3X_3$	${}_4X_3$	${}_5X_3$	${}_6X_3$	${}_7X_3$	${}_8\hat{X}_3$	${}_9\hat{X}_3$
4	${}_0X_4$	${}_1X_4$	${}_2X_4$	${}_3X_4$	${}_4X_4$	${}_5X_4$	${}_6X_4$	${}_7\hat{X}_4$	${}_8\hat{X}_4$	${}_9\hat{X}_4$
5	${}_0X_5$	${}_1X_5$	${}_2X_5$	${}_3X_5$	${}_4X_5$	${}_5X_5$	${}_6\hat{X}_5$	${}_7\hat{X}_5$	${}_8\hat{X}_5$	${}_9\hat{X}_5$
6	${}_0X_6$	${}_1X_6$	${}_2X_6$	${}_3X_6$	${}_4X_6$	${}_5\hat{X}_6$	${}_6\hat{X}_6$	${}_7\hat{X}_6$	${}_8\hat{X}_6$	${}_9\hat{X}_6$
7	${}_0X_7$	${}_1X_7$	${}_2X_7$	${}_3X_7$	${}_4\hat{X}_7$	${}_5\hat{X}_7$	${}_6\hat{X}_7$	${}_7\hat{X}_7$	${}_8\hat{X}_7$	${}_9\hat{X}_7$
8	${}_0X_8$	${}_1X_8$	${}_2X_8$	${}_3\hat{X}_8$	${}_4\hat{X}_8$	${}_5\hat{X}_8$	${}_6\hat{X}_8$	${}_7\hat{X}_8$	${}_8\hat{X}_8$	${}_9\hat{X}_8$
9	${}_0X_9$	${}_1X_9$	${}_2\hat{X}_9$	${}_3\hat{X}_9$	${}_4\hat{X}_9$	${}_5\hat{X}_9$	${}_6\hat{X}_9$	${}_7\hat{X}_9$	${}_8\hat{X}_9$	${}_9\hat{X}_9$
10	${}_0X_{10}$	${}_1\hat{X}_{10}$	${}_2\hat{X}_{10}$	${}_3\hat{X}_{10}$	${}_4\hat{X}_{10}$	${}_5\hat{X}_{10}$	${}_6\hat{X}_{10}$	${}_7\hat{X}_{10}$	${}_8\hat{X}_{10}$	${}_9\hat{X}_{10}$

Table 2.4: An illustration of prediction of the `X1`.

	0	1	2	3	4	5	6	7	8	9
1	${}_0X_1$	${}_1X_1$	${}_2X_1$	${}_3X_1$	${}_4X_1$	${}_5X_1$	${}_6X_1$	${}_7X_1$	${}_8X_1$	${}_9X_1$
2	${}_0X_2$	${}_1X_2$	${}_2X_2$	${}_3X_2$	${}_4X_2$	${}_5X_2$	${}_6X_2$	${}_7X_2$	${}_8X_2$	${}_9\hat{X}_2$
3	${}_0X_3$	${}_1X_3$	${}_2X_3$	${}_3X_3$	${}_4X_3$	${}_5X_3$	${}_6X_3$	${}_7X_3$	${}_8\hat{X}_3$	${}_9\hat{X}_3$
4	${}_0X_4$	${}_1X_4$	${}_2X_4$	${}_3X_4$	${}_4X_4$	${}_5X_4$	${}_6X_4$	${}_7\hat{X}_4$	${}_8\hat{X}_4$	${}_9\hat{X}_4$
5	${}_0X_5$	${}_1X_5$	${}_2X_5$	${}_3X_5$	${}_4X_5$	${}_5X_5$	${}_6\hat{X}_5$	${}_7\hat{X}_5$	${}_8\hat{X}_5$	${}_9\hat{X}_5$
6	${}_0X_6$	${}_1X_6$	${}_2X_6$	${}_3X_6$	${}_4X_6$	${}_5\hat{X}_6$	${}_6\hat{X}_6$	${}_7\hat{X}_6$	${}_8\hat{X}_6$	${}_9\hat{X}_6$
7	${}_0X_7$	${}_1X_7$	${}_2X_7$	${}_3X_7$	${}_4\hat{X}_7$	${}_5\hat{X}_7$	${}_6\hat{X}_7$	${}_7\hat{X}_7$	${}_8\hat{X}_7$	${}_9\hat{X}_7$
8	${}_0X_8$	${}_1X_8$	${}_2X_8$	${}_3\hat{X}_8$	${}_4\hat{X}_8$	${}_5\hat{X}_8$	${}_6\hat{X}_8$	${}_7\hat{X}_8$	${}_8\hat{X}_8$	${}_9\hat{X}_8$
9	${}_0X_9$	${}_1X_9$	${}_2\hat{X}_9$	${}_3\hat{X}_9$	${}_4\hat{X}_9$	${}_5\hat{X}_9$	${}_6\hat{X}_9$	${}_7\hat{X}_9$	${}_8\hat{X}_9$	${}_9\hat{X}_9$
10	${}_0X_{10}$	${}_1\hat{X}_{10}$	${}_2\hat{X}_{10}$	${}_3\hat{X}_{10}$	${}_4\hat{X}_{10}$	${}_5\hat{X}_{10}$	${}_6\hat{X}_{10}$	${}_7\hat{X}_{10}$	${}_8\hat{X}_{10}$	${}_9\hat{X}_{10}$

Table 2.5: An illustration of the situation after the prediction of `X1`.

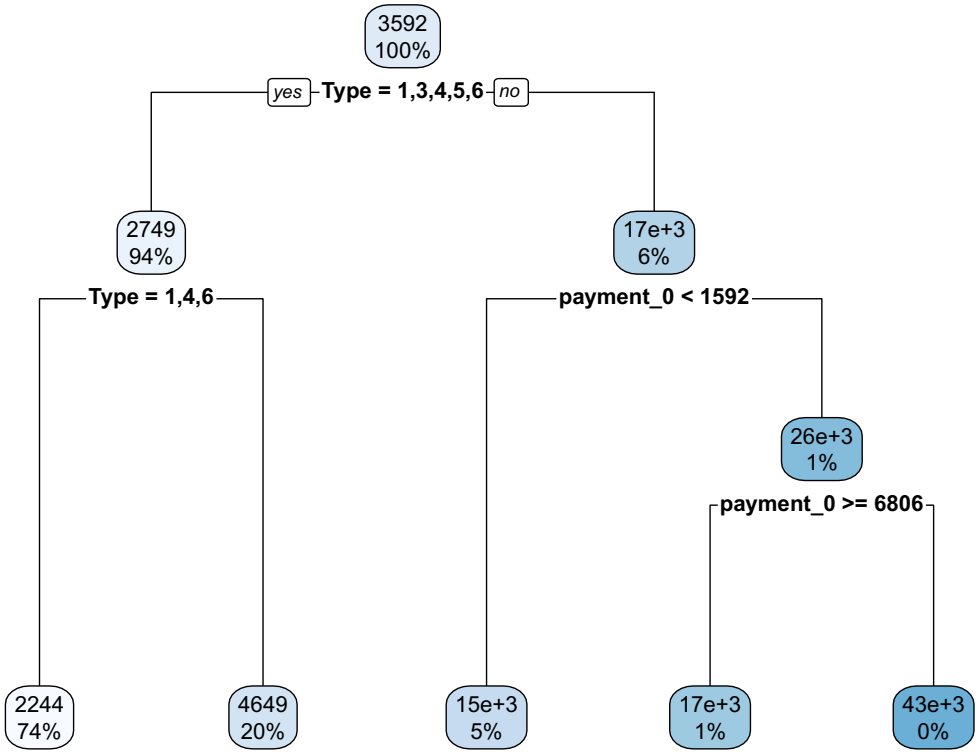


Figure 2.3: Optimal tree for $l = 0$.

Then for the development period $l = d + k = 1$ we will use information $X^{(v)}_{i,1\,|\,0+1}$, $\mathbf{X}^{(v)}_{i,1\,|\,0}$, $X^{(v)}_{i,0\,|\,1+1}$, and $\mathbf{X}^{(v)}_{i,0\,|\,1}$. That means we will regress `X2` as the response from variables `Type, Age, weekday, d, X0, X1`. As it has to hold $i + d + k + 1 \leq 10$ and $d \in \{0, 1\}$ that means we can use data such that $i \leq 8$. This approach is again depicted in Table 2.6, where the known values of `X2` are in lighter green and `X1` and `X0` are in darker green.

	0	1	2	3	4	5	6	7	8	9
1	${}_0X_1$	${}_1X_1$	${}_2X_1$	${}_3X_1$	${}_4X_1$	${}_5X_1$	${}_6X_1$	${}_7X_1$	${}_8X_1$	${}_9X_1$
2	${}_0X_2$	${}_1X_2$	${}_2X_2$	${}_3X_2$	${}_4X_2$	${}_5X_2$	${}_6X_2$	${}_7X_2$	${}_8X_2$	${}_9\hat{X}_2$
3	${}_0X_3$	${}_1X_3$	${}_2X_3$	${}_3X_3$	${}_4X_3$	${}_5X_3$	${}_6X_3$	${}_7X_3$	${}_8\hat{X}_3$	${}_9\hat{X}_3$
4	${}_0X_4$	${}_1X_4$	${}_2X_4$	${}_3X_4$	${}_4X_4$	${}_5X_4$	${}_6X_4$	${}_7\hat{X}_4$	${}_8\hat{X}_4$	${}_9\hat{X}_4$
5	${}_0X_5$	${}_1X_5$	${}_2X_5$	${}_3X_5$	${}_4X_5$	${}_5X_5$	${}_6\hat{X}_5$	${}_7\hat{X}_5$	${}_8\hat{X}_5$	${}_9\hat{X}_5$
6	${}_0X_6$	${}_1X_6$	${}_2X_6$	${}_3X_6$	${}_4X_6$	${}_5\hat{X}_6$	${}_6\hat{X}_6$	${}_7\hat{X}_6$	${}_8\hat{X}_6$	${}_9\hat{X}_6$
7	${}_0X_7$	${}_1X_7$	${}_2X_7$	${}_3X_7$	${}_4\hat{X}_7$	${}_5\hat{X}_7$	${}_6\hat{X}_7$	${}_7\hat{X}_7$	${}_8\hat{X}_7$	${}_9\hat{X}_7$
8	${}_0X_8$	${}_1X_8$	${}_2X_8$	${}_3\hat{X}_8$	${}_4\hat{X}_8$	${}_5\hat{X}_8$	${}_6\hat{X}_8$	${}_7\hat{X}_8$	${}_8\hat{X}_8$	${}_9\hat{X}_8$
9	${}_0X_9$	${}_1X_9$	${}_2\hat{X}_9$	${}_3\hat{X}_9$	${}_4\hat{X}_9$	${}_5\hat{X}_9$	${}_6\hat{X}_9$	${}_7\hat{X}_9$	${}_8\hat{X}_9$	${}_9\hat{X}_9$
10	${}_0X_{10}$	${}_1\hat{X}_{10}$	${}_2\hat{X}_{10}$	${}_3\hat{X}_{10}$	${}_4\hat{X}_{10}$	${}_5\hat{X}_{10}$	${}_6\hat{X}_{10}$	${}_7\hat{X}_{10}$	${}_8\hat{X}_{10}$	${}_9\hat{X}_{10}$

Table 2.6: An illustration of how we learn to predict `X2`.

Now that our method has learned how to predict `X2`, we can apply this and predict ${}_2\hat{X}_9$ and ${}_2\hat{X}_{10}$. This is illustrated in Table 2.7, where we predict the values in light yellow based on the values in darker yellow. However, here comes one small adjustment, we also include the value ${}_1\hat{\hat{X}}_{10}$ that we predicted in the first step to the covariates vector and thus make a prediction based on this value. We have now predicted the values ${}_2\hat{X}_9$ and ${}_2\hat{X}_{10}$.

	0	1	2	3	4	5	6	7	8	9
1	${}_0X_1$	${}_1X_1$	${}_2X_1$	${}_3X_1$	${}_4X_1$	${}_5X_1$	${}_6X_1$	${}_7X_1$	${}_8X_1$	${}_9X_1$
2	${}_0X_2$	${}_1X_2$	${}_2X_2$	${}_3X_2$	${}_4X_2$	${}_5X_2$	${}_6X_2$	${}_7X_2$	${}_8X_2$	${}_9\hat{X}_2$
3	${}_0X_3$	${}_1X_3$	${}_2X_3$	${}_3X_3$	${}_4X_3$	${}_5X_3$	${}_6X_3$	${}_7X_3$	${}_8\hat{X}_3$	${}_9\hat{X}_3$
4	${}_0X_4$	${}_1X_4$	${}_2X_4$	${}_3X_4$	${}_4X_4$	${}_5X_4$	${}_6X_4$	${}_7\hat{X}_4$	${}_8\hat{X}_4$	${}_9\hat{X}_4$
5	${}_0X_5$	${}_1X_5$	${}_2X_5$	${}_3X_5$	${}_4X_5$	${}_5X_5$	${}_6\hat{X}_5$	${}_7\hat{X}_5$	${}_8\hat{X}_5$	${}_9\hat{X}_5$
6	${}_0X_6$	${}_1X_6$	${}_2X_6$	${}_3X_6$	${}_4X_6$	${}_5\hat{X}_6$	${}_6\hat{X}_6$	${}_7\hat{X}_6$	${}_8\hat{X}_6$	${}_9\hat{X}_6$
7	${}_0X_7$	${}_1X_7$	${}_2X_7$	${}_3X_7$	${}_4\hat{X}_7$	${}_5\hat{X}_7$	${}_6\hat{X}_7$	${}_7\hat{X}_7$	${}_8\hat{X}_7$	${}_9\hat{X}_7$
8	${}_0X_8$	${}_1X_8$	${}_2X_8$	${}_3\hat{X}_8$	${}_4\hat{X}_8$	${}_5\hat{X}_8$	${}_6\hat{X}_8$	${}_7\hat{X}_8$	${}_8\hat{X}_8$	${}_9\hat{X}_8$
9	${}_0X_9$	${}_1X_9$	${}_2\hat{X}_9$	${}_3\hat{X}_9$	${}_4\hat{X}_9$	${}_5\hat{X}_9$	${}_6\hat{X}_9$	${}_7\hat{X}_9$	${}_8\hat{X}_9$	${}_9\hat{X}_9$
10	${}_0X_{10}$	${}_1\hat{X}_{10}$	${}_2\hat{X}_{10}$	${}_3\hat{X}_{10}$	${}_4\hat{X}_{10}$	${}_5\hat{X}_{10}$	${}_6\hat{X}_{10}$	${}_7\hat{X}_{10}$	${}_8\hat{X}_{10}$	${}_9\hat{X}_{10}$

Table 2.7: An illustration of prediction of `X2`.

This continues analogously until $l = d + k = 8$. The last learning process is depicted in Table 2.8, where we can use only the data for accident year $i = 1$ as we do not have any information for claims with accident years greater than 1 about their development for $l = 9$. We will use information $X^{(v)}_{i,8\,|\,0+1}$, $\mathbf{X}^{(v)}_{i,8\,|\,0}$, $X^{(v)}_{i,7\,|\,1+1}$, $\mathbf{X}^{(v)}_{i,7\,|\,1}, \dots X^{(v)}_{i,1\,|\,0+1}$, $\mathbf{X}^{(v)}_{i,1\,|\,0}$, $X^{(v)}_{i,0\,|\,8+1}$, and $\mathbf{X}^{(v)}_{i,0\,|\,8}$. That means we will regress `X9` as the response from variables `Type, Age, weekday, d, X0, X1, X2, X3, X4, X5, X6, X7, X8`. This approach is again depicted in Table 2.8, where the known value of `X9` is in lighter green and the corresponding values of `X8, X7, X6, X5, X4, X3, X2, X1` and `X0` are in darker green.

Now that our method has learned how to predict `X9`, we can apply this and predict ${}_9\hat{X}_2, {}_9\hat{X}_3, \dots, {}_9\hat{X}_{10}$. This is illustrated in Table 2.9, where we predict the values in light yellow based on the values in darker yellow and again, we will use the unknown values `X8, X7, X6, X5, X4, X3, X2, X1` we predicted in the previous steps.

2.7.5 Prediction for IBNR claims

In the previous section, we discussed the sums of payments for reported claims. Now we aim to predict IBNR claims. These claims are given by

$$\mathcal{F}_t^{IBNR} = \left\{\mathbf{x}^{(v)}_{i,d\,|\,k}, i + d + k > t,\ v \le N_{i,d},\ i + d > t\right\}.$$

The total numbers of reported claims $N_{i,d}$ for $i + d > 10$ are not yet observed. Thus we also need to predict this quantity. For this, we need another model assumption. If we assume that the claims occurrence and reporting process can be

	0	1	2	3	4	5	6	7	8	9
1	${}_0X_1$	${}_1X_1$	${}_2X_1$	${}_3X_1$	${}_4X_1$	${}_5X_1$	${}_6X_1$	${}_7X_1$	${}_8X_1$	${}_9X_1$
2	${}_0X_2$	${}_1X_2$	${}_2X_2$	${}_3X_2$	${}_4X_2$	${}_5X_2$	${}_6X_2$	${}_7X_2$	${}_8X_2$	${}_9\hat{X}_2$
3	${}_0X_3$	${}_1X_3$	${}_2X_3$	${}_3X_3$	${}_4X_3$	${}_5X_3$	${}_6X_3$	${}_7X_3$	${}_8\hat{X}_3$	${}_9\hat{X}_3$
4	${}_0X_4$	${}_1X_4$	${}_2X_4$	${}_3X_4$	${}_4X_4$	${}_5X_4$	${}_6X_4$	${}_7\hat{X}_4$	${}_8\hat{X}_4$	${}_9\hat{X}_4$
5	${}_0X_5$	${}_1X_5$	${}_2X_5$	${}_3X_5$	${}_4X_5$	${}_5X_5$	${}_6\hat{X}_5$	${}_7\hat{X}_5$	${}_8\hat{X}_5$	${}_9\hat{X}_5$
6	${}_0X_6$	${}_1X_6$	${}_2X_6$	${}_3X_6$	${}_4X_6$	${}_5\hat{X}_6$	${}_6\hat{X}_6$	${}_7\hat{X}_6$	${}_8\hat{X}_6$	${}_9\hat{X}_6$
7	${}_0X_7$	${}_1X_7$	${}_2X_7$	${}_3X_7$	${}_4\hat{X}_7$	${}_5\hat{X}_7$	${}_6\hat{X}_7$	${}_7\hat{X}_7$	${}_8\hat{X}_7$	${}_9\hat{X}_7$
8	${}_0X_8$	${}_1X_8$	${}_2X_8$	${}_3\hat{X}_8$	${}_4\hat{X}_8$	${}_5\hat{X}_8$	${}_6\hat{X}_8$	${}_7\hat{X}_8$	${}_8\hat{X}_8$	${}_9\hat{X}_8$
9	${}_0X_9$	${}_1X_9$	${}_2\hat{X}_9$	${}_3\hat{X}_9$	${}_4\hat{X}_9$	${}_5\hat{X}_9$	${}_6\hat{X}_9$	${}_7\hat{X}_9$	${}_8\hat{X}_9$	${}_9\hat{X}_9$
10	${}_0X_{10}$	${}_1\hat{X}_{10}$	${}_2\hat{X}_{10}$	${}_3\hat{X}_{10}$	${}_4\hat{X}_{10}$	${}_5\hat{X}_{10}$	${}_6\hat{X}_{10}$	${}_7\hat{X}_{10}$	${}_8\hat{X}_{10}$	${}_9\hat{X}_{10}$

Table 2.8: An illustration of how we learn to predict `X9`.

described by a homogeneous marked Poisson point process, then these numbers of reported claims are in line with the classical chain-ladder method. This is discussed in more detail in Section 6.1 in Verrall and Wüthrich [2016]. The corresponding predictions for $N_{i,d}$ are then easily obtained using the chain-ladder method.

If, in addition to Assumption 1, we assume that the payments

$$\left(X^{(v)}_{i,d\,|\,0}, X^{(v)}_{i,d\,|\,1}, \ldots, X^{(v)}_{i,d\,|\,L-d}\right)$$

are independent of $\mathcal{F}_t$ for $i+d>t$, then we can predict the quantity

$${}_l\hat{X}_i^{IBNR},$$

which represents the prediction of the sum of all payments (i,d,k,v), such that $d+k=l$ and $i+d+k=i+l>I$ for claims (i,d,v), such that $i+d>I$ with

	0	1	2	3	4	5	6	7	8	9
1	${}_0X_1$	${}_1X_1$	${}_2X_1$	${}_3X_1$	${}_4X_1$	${}_5X_1$	${}_6X_1$	${}_7X_1$	${}_8X_1$	${}_9X_1$
2	${}_0X_2$	${}_1X_2$	${}_2X_2$	${}_3X_2$	${}_4X_2$	${}_5X_2$	${}_6X_2$	${}_7X_2$	${}_8X_2$	${}_9\hat{X}_2$
3	${}_0X_3$	${}_1X_3$	${}_2X_3$	${}_3X_3$	${}_4X_3$	${}_5X_3$	${}_6X_3$	${}_7X_3$	${}_8\hat{X}_3$	${}_9\hat{X}_3$
4	${}_0X_4$	${}_1X_4$	${}_2X_4$	${}_3X_4$	${}_4X_4$	${}_5X_4$	${}_6X_4$	${}_7\hat{X}_4$	${}_8\hat{X}_4$	${}_9\hat{X}_4$
5	${}_0X_5$	${}_1X_5$	${}_2X_5$	${}_3X_5$	${}_4X_5$	${}_5X_5$	${}_6\hat{X}_5$	${}_7\hat{X}_5$	${}_8\hat{X}_5$	${}_9\hat{X}_5$
6	${}_0X_6$	${}_1X_6$	${}_2X_6$	${}_3X_6$	${}_4X_6$	${}_5\hat{X}_6$	${}_6\hat{X}_6$	${}_7\hat{X}_6$	${}_8\hat{X}_6$	${}_9\hat{X}_6$
7	${}_0X_7$	${}_1X_7$	${}_2X_7$	${}_3X_7$	${}_4\hat{X}_7$	${}_5\hat{X}_7$	${}_6\hat{X}_7$	${}_7\hat{X}_7$	${}_8\hat{X}_7$	${}_9\hat{X}_7$
8	${}_0X_8$	${}_1X_8$	${}_2X_8$	${}_3\hat{X}_8$	${}_4\hat{X}_8$	${}_5\hat{X}_8$	${}_6\hat{X}_8$	${}_7\hat{X}_8$	${}_8\hat{X}_8$	${}_9\hat{X}_8$
9	${}_0X_9$	${}_1X_9$	${}_2\hat{X}_9$	${}_3\hat{X}_9$	${}_4\hat{X}_9$	${}_5\hat{X}_9$	${}_6\hat{X}_9$	${}_7\hat{X}_9$	${}_8\hat{X}_9$	${}_9\hat{X}_9$
10	${}_0X_{10}$	${}_1\hat{X}_{10}$	${}_2\hat{X}_{10}$	${}_3\hat{X}_{10}$	${}_4\hat{X}_{10}$	${}_5\hat{X}_{10}$	${}_6\hat{X}_{10}$	${}_7\hat{X}_{10}$	${}_8\hat{X}_{10}$	${}_9\hat{X}_{10}$

Table 2.9: An illustration of prediction of `X9`.

accident year i being paid in year $i+l$ by the following formula:

$$ {}_l\hat{X}_i^{IBNR} = \hat{N}_{i,l}^{(cCL)} \hat{\mathbb{E}}[{}_lX_i^{(1)}], $$

where $\hat{N}_{i,l}^{(cCL)}$ is the chain-ladder estimation of the cumulative number of claims calculated from the reported claims. This represents the number of claims with accident year i that have been reported up to the time $i+d$. The quantity

$$ \hat{\mathbb{E}}[{}_lX_i^{(1)}] $$

is the expected value of the payment $(i,d,k,1)$ such that $d+k=l$ for a single claim (i,d,v), such that $i+d>I$ with accident year i.

Assuming homogeneity in the accident years $i \in \{1,2,\dots,I\}$ then for $i \in \{1,2,\dots,I\}$ it holds

$$ \hat{\mathbb{E}}[{}_lX_i^{(1)}] = \hat{\mathbb{E}}[{}_lX^{(1)}], $$

which is the expected value of a payment (i,d,k,v) such that $d+k=l$ for a single claim (i,d,v), such that $i+d>I$. This quantity is then easily estimated via the average

$$ \frac{1}{\sum\limits_{i=1}^{I} N_{i,l}^{(c)}} \sum_{i=1}^{I} {}_l\hat{X}_i, $$

where $N_{i,d}^{(c)} = \sum\limits_{j=1}^{d} N_{i,j}$, i.e., it is the number of reported claims with accident year i up to time $i+d$, i.e., the cumulative number of claims which represents how many claims with accident year i have been reported so far.

Then it is easy to calculate for $l>I$

$$ {}_l\hat{X}^{IBNR} = \sum_{i=\max\{1,l-L\}}^{\min\{l,I\}} {}_{l-i}\hat{X}_i^{IBNR}, $$

and

$$ \hat{X}^{IBNR} = \sum_{l=I+1}^{2I-1} {}_l\hat{X}^{IBNR}. $$

Overall, the quantity

$$ {}_l\hat{X}_i^{IBNR} $$

represents the prediction of the sum of all payments (i,d,k,v), such that $d+k=l$ and $i+d+k=i+l>I$ for claims (i,d,v), such that $i+d>I$ with accident year i, being paid in year $i+l$.

The quantity

$$ {}_l\hat{X}^{IBNR} $$

represents the prediction of the sum of all payments (i,d,k,v), such that $d+k=l$, and $i+d+k=i+l>I$ for claims (i,d,v), such that $i+d>I$, being paid in year l. Lastly, the quantity

$$ \hat{X}^{IBNR} $$

represents the prediction of the sum of payments (i,d,k,v), such that $i+d+k>I$ for claims (i,d,v), such that $i+d>I$.

2.8 Numerical analysis

2.8.1 Our predictions

At the time $t = I = 10$ we have information about claims (i, d, v), such that $i + d \leq 10$ and about payments (i, d, k, v), such that $i + d + k \leq 10$. This is illustrated in Table 2.10, where in rows there are accident years $i \in \{1, 2, \ldots, 10\}$. And in columns there are development delays $l \in \{0, 1 \ldots, 9\}$. Then the cell (i, l) represents the sum of all payments with accident years i paid in year $i + l$. As we mentioned before, our goal is to predict the payments ${}_l\hat{X}_i$ such that $i + l > 10$, i.e., those values in Table 2.10 that are in red.

	0	1	2	3	4	5	6	7	8	9
1	${}_0X_1$	${}_1X_1$	${}_2X_1$	${}_3X_1$	${}_4X_1$	${}_5X_1$	${}_6X_1$	${}_7X_1$	${}_8X_1$	${}_9X_1$
2	${}_0X_2$	${}_1X_2$	${}_2X_2$	${}_3X_2$	${}_4X_2$	${}_5X_2$	${}_6X_2$	${}_7X_2$	${}_8X_2$	${}_9\hat{X}_2$
3	${}_0X_3$	${}_1X_3$	${}_2X_3$	${}_3X_3$	${}_4X_3$	${}_5X_3$	${}_6X_3$	${}_7X_3$	${}_8\hat{X}_3$	${}_9\hat{X}_3$
4	${}_0X_4$	${}_1X_4$	${}_2X_4$	${}_3X_4$	${}_4X_4$	${}_5X_4$	${}_6X_4$	${}_7\hat{X}_4$	${}_8\hat{X}_4$	${}_9\hat{X}_4$
5	${}_0X_5$	${}_1X_5$	${}_2X_5$	${}_3X_5$	${}_4X_5$	${}_5X_5$	${}_6\hat{X}_5$	${}_7\hat{X}_5$	${}_8\hat{X}_5$	${}_9\hat{X}_5$
6	${}_0X_6$	${}_1X_6$	${}_2X_6$	${}_3X_6$	${}_4X_6$	${}_5\hat{X}_6$	${}_6\hat{X}_6$	${}_7\hat{X}_6$	${}_8\hat{X}_6$	${}_9\hat{X}_6$
7	${}_0X_7$	${}_1X_7$	${}_2X_7$	${}_3X_7$	${}_4\hat{X}_7$	${}_5\hat{X}_7$	${}_6\hat{X}_7$	${}_7\hat{X}_7$	${}_8\hat{X}_7$	${}_9\hat{X}_7$
8	${}_0X_8$	${}_1X_8$	${}_2X_8$	${}_3\hat{X}_8$	${}_4\hat{X}_8$	${}_5\hat{X}_8$	${}_6\hat{X}_8$	${}_7\hat{X}_8$	${}_8\hat{X}_8$	${}_9\hat{X}_8$
9	${}_0X_9$	${}_1X_9$	${}_2\hat{X}_9$	${}_3\hat{X}_9$	${}_4\hat{X}_9$	${}_5\hat{X}_9$	${}_6\hat{X}_9$	${}_7\hat{X}_9$	${}_8\hat{X}_9$	${}_9\hat{X}_9$
10	${}_0X_{10}$	${}_1\hat{X}_{10}$	${}_2\hat{X}_{10}$	${}_3\hat{X}_{10}$	${}_4\hat{X}_{10}$	${}_5\hat{X}_{10}$	${}_6\hat{X}_{10}$	${}_7\hat{X}_{10}$	${}_8\hat{X}_{10}$	${}_9\hat{X}_{10}$

Table 2.10: An illustration of our run-off situation.

If we fill in the numerical values we generated, we get Table 2.11. The values to be predicted are in the red cells, and they are labeled ${}_l\hat{X}_i$, where i indicates the accident year and l indicates the payment l years after the accident year.

	0	1	2	3	4	5	6	7	8	9
1	5106	18504	19393	10596	6231	3239	1798	685	174	58
2	4578	19141	18642	11188	5586	2833	1415	246	238	${}_9\hat{X}_2$
3	5162	19699	17175	10372	5766	2368	580	347	${}_8\hat{X}_3$	${}_9\hat{X}_3$
4	3801	16529	15064	7587	3827	2306	674	${}_7\hat{X}_4$	${}_8\hat{X}_4$	${}_9\hat{X}_4$
5	4117	15932	14432	8564	2932	2772	${}_6\hat{X}_5$	${}_7\hat{X}_5$	${}_8\hat{X}_5$	${}_9\hat{X}_5$
6	3703	16250	12681	8287	3935	${}_5\hat{X}_6$	${}_6\hat{X}_6$	${}_7\hat{X}_6$	${}_8\hat{X}_6$	${}_9\hat{X}_6$
7	3839	16012	13974	7116	${}_4\hat{X}_7$	${}_5\hat{X}_7$	${}_6\hat{X}_7$	${}_7\hat{X}_7$	${}_8\hat{X}_7$	${}_9\hat{X}_7$
8	4425	16565	14693	${}_3\hat{X}_8$	${}_4\hat{X}_8$	${}_5\hat{X}_8$	${}_6\hat{X}_8$	${}_7\hat{X}_8$	${}_8\hat{X}_8$	${}_9\hat{X}_8$
9	3994	16425	${}_2\hat{X}_9$	${}_3\hat{X}_9$	${}_4\hat{X}_9$	${}_5\hat{X}_9$	${}_6\hat{X}_9$	${}_7\hat{X}_9$	${}_8\hat{X}_9$	${}_9\hat{X}_9$
10	5067	${}_1\hat{X}_{10}$	${}_2\hat{X}_{10}$	${}_3\hat{X}_{10}$	${}_4\hat{X}_{10}$	${}_5\hat{X}_{10}$	${}_6\hat{X}_{10}$	${}_7\hat{X}_{10}$	${}_8\hat{X}_{10}$	${}_9\hat{X}_{10}$

Table 2.11: The run-off situation for generated data (in thousands).

In Table 2.12, we see what the real payments looked like. (We know these data as the data generator from Wang and Wüthrich [2022] also produces the payments after the time $I = 10$.)

	0	1	2	3	4	5	6	7	8	9
1	5106	18504	19393	10596	6231	3239	1798	685	174	58
2	4578	19141	18642	11188	5586	2833	115	246	238	122
3	5162	19699	17175	10372	5766	2368	580	347	292	−83
4	3801	16529	15064	7587	3827	2306	674	118	−51	74
5	4117	15932	14432	8564	2932	2772	801	364	115	−5
6	3703	16250	12681	8287	3935	1471	942	268	182	77
7	3839	16012	13974	7116	3557	1609	756	314	15	27
8	4425	16565	14693	8148	2889	1254	260	367	249	93
9	3994	16425	13685	7823	3353	1032	1444	308	43	214
10	5067	16197	15631	7531	3835	2485	1005	190	57	29

Table 2.12: The full table with real payments that we want to predict (in thousands).

It is interesting that there are even some negative values. This indicates that there was more money given back to the insurance company than paid for claims. Table 2.13 presents the estimates of outstanding loss liabilities calculated using the Chain-ladder method.

	0	1	2	3	4	5	6	7	8	9
1	5106	18504	19393	10596	6231	3239	1798	685	174	58
2	4578	19141	18642	11188	5586	2833	1415	246	238	80
3	5162	19699	17175	10372	5766	2368	580	347	153	52
4	3801	16529	15064	7587	3827	2306	674	227	100	34
5	4117	15932	14432	8564	2932	2772	1152	388	172	58
6	3703	16250	12681	8287	3935	2185	908	306	135	45
7	3839	16012	13974	7116	3556	1975	821	277	122	41
8	4425	16565	14693	8406	4200	2332	969	327	144	48
9	3994	16425	14935	8544	4269	2371	985	332	147	49
10	5067	20289	18448	10554	5273	2929	1217	410	181	61

Table 2.13: The table generated by chain-ladder method (in thousands).

In Table 2.14, we present the estimates of outstanding loss liabilities calculated using regression trees. The first striking observation is that for the development delay $l = 6$, there are very high values in the table. This is probably caused by overfitting, and it very much affects the reserves calculated. In Table 2.15 we can see reserves calculated via the bagging method for 10000 bags. We can see that the values look much more reasonable.

2.8.2 The results

In this part of the thesis, we present the results we obtained. A lot of work is behind this. However, it cannot be seen here, as it lies in the programming part in `R` that we attach together with this thesis.

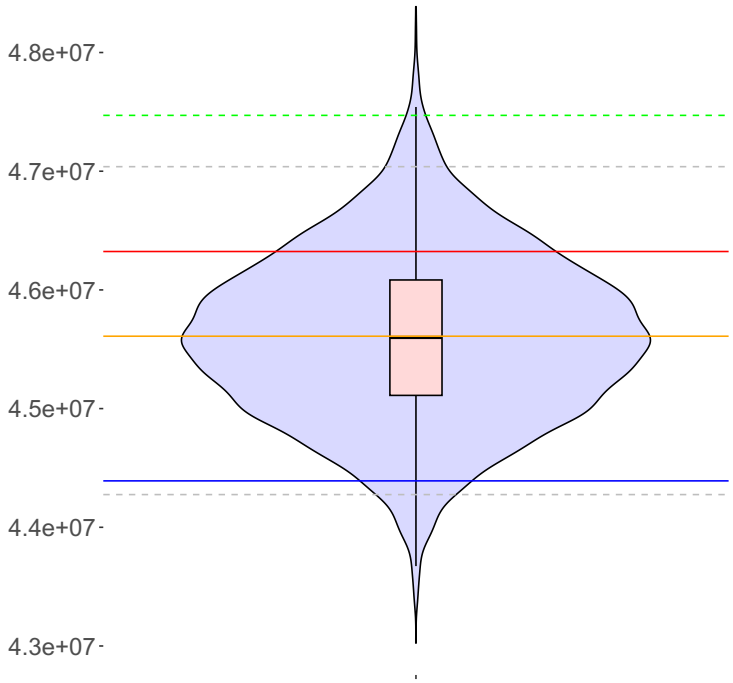


Figure 2.4: The violin plot of next year's reserves. The blue horizontal line represents the value of real payments, the red line represents the estimate of payments calculated via the regression trees, and the orange line represents the estimate of payments calculated via the bagging method. The grey dashed lines represent the 95% confidence interval, and the green dashed line represents the 99.5% quantile.

	0	1	2	3	4	5	6	7	8	9
1	5106	18504	19393	10596	6231	3239	1798	685	174	58
2	4578	19141	18642	11188	5586	2833	1415	246	238	58
3	5162	19699	17175	10372	5766	2368	580	347	219	62
4	3801	16529	15064	7587	3827	2306	674	413	203	57
5	4117	15932	14432	8564	2932	2772	711	415	205	58
6	3703	16250	12681	8287	3935	2473	34090	381	188	53
7	3839	16012	13974	7116	4138	4064	25239	363	179	50
8	4425	16565	14693	8257	3220	6802	20507	350	173	49
9	3994	16425	12876	7026	3188	4973	17449	334	165	46
10	5067	17179	10978	6954	3250	1959	24070	348	172	48

Table 2.14: The table generated by optimal regression trees (in thousands).

	0	1	2	3	4	5	6	7	8	9
1	5106	18504	19393	10596	6231	3239	1798	685	174	58
2	4578	19141	18642	11188	5586	2833	1415	246	238	55
3	5162	19699	17175	10372	5766	2368	580	347	159	49
4	3801	16529	15064	7587	3827	2306	674	300	676	31
5	4117	15932	14432	8564	2932	2772	665	970	821	31
6	3703	16250	12681	8287	3935	2028	2991	1484	1018	26
7	3839	16012	13974	7116	3875	4412	3157	1617	1107	52
8	4425	16565	14693	7673	4324	5514	3132	1605	1116	72
9	3994	16425	13369	6394	3995	5024	2775	1272	981	74
10	5067	17485	11651	7033	3670	4475	2035	1055	832	65

Table 2.15: The table generated by bagging method (in thousands).

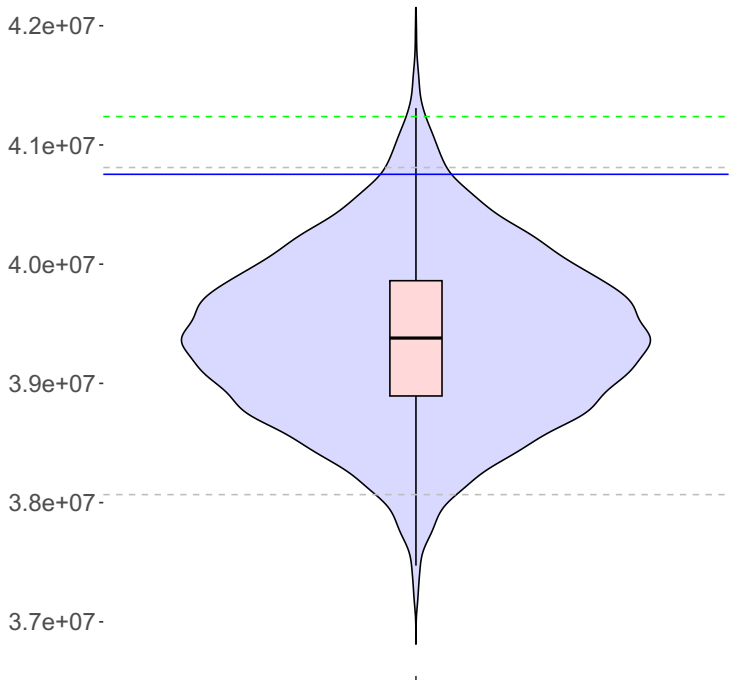


Figure 2.5: The violin plot of next year's payments for RBNS claims. The blue line represents the value of actual payments. The grey dashed lines represent the 95% confidence interval while the green dashed line represents the 99.5% quantile.

More than the sum of payments for claims with accident years i paid in year $i + l$, we are interested in two aggregated quantities: the sum of payments for next year that was in our case denoted by ${}_{11}X$ and the sum of all payments in the future that can be calculated by the sum ${}_{11}X + {}_{12}X + \ldots + {}_{19}X$ and their estimates denoted by ${}_{11}\hat{X}$ and ${}_{11}\hat{X} + {}_{12}\hat{X} + \ldots + {}_{19}\hat{X}$. Thanks to the bagging approach, we have empirical distribution for (not only) these quantities.

Next year's reserves

In Figure 2.4, we can see a violin plot of the next year's payments together with the actual value of next year's payments, represented by the horizontal blue line. The red line in the plot represents the next year's payment estimation obtained from the regression tree. The orange line represents the estimate of next year's payments received by the bagging method. The grey dashed lines represent the 95% confidence interval, and the green dashed line represents the 99.5% quantile.

We can clearly see that both the bagging method and the tree method overestimate the sum of the next year's payments. For comparison, the chain ladder method estimates the next payments even worse by more than $5 \cdot 10^7$, which is so much that it would not fit into our plot.

We tried to diagnose the cause of this trouble. In Figure 2.5 presents the estimates for next year's RBNS claims and Figure 2.6 for next year's IBNR claims. We can see in the first figure that the bagging method underestimates the reserves needed for the RBNS claim. While in the second figure, we do not even have the blue line representing the real value for IBNR payments because it is less than $37 \cdot 10^5$, while the values obtained by the bagging method range around $62 \cdot 10^5$. Thus, we can conclude that the problem with overestimating the next year's reserves is caused by the IBNR claims.

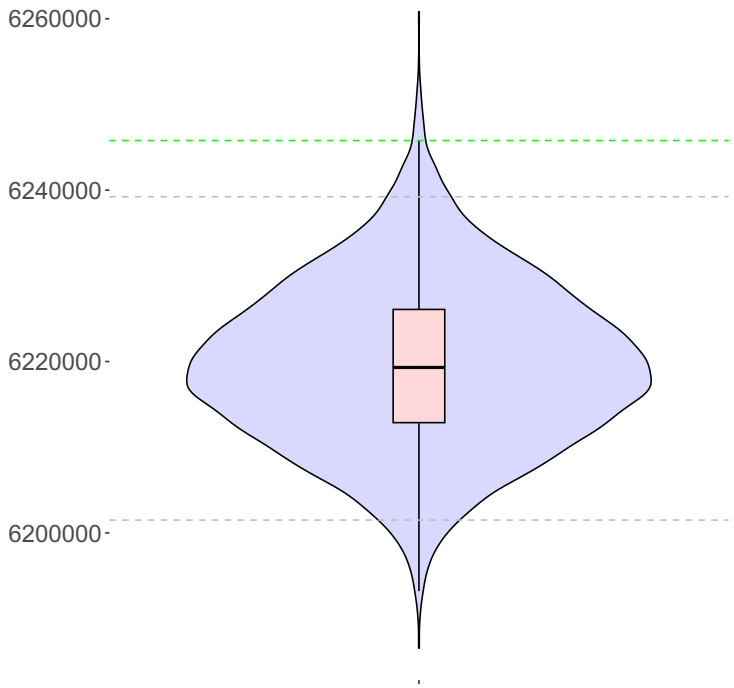


Figure 2.6: The violin plot of next year's payments for IBNR claims. The value of real payments is not in the figure as it is too low. The grey dashed lines represent the 95% confidence interval while the green dashed line represents the 99.5% quantile.

Ultimate reserves

Figure 2.7 displays the violin plot of the sum of all future payments together with the actual value of the sum of all future reserves represented by the blue horizontal line. The green horizontal line represents the estimate from the chain ladder method, and the orange line represents the estimate from the bagging approach. We can again see that the bagging method overestimates the total sum of payments. The chain ladder method performs slightly better. For comparison, the tree model predicted values larger than $26 \cdot 10^7$, which is again out of the graph's scale.

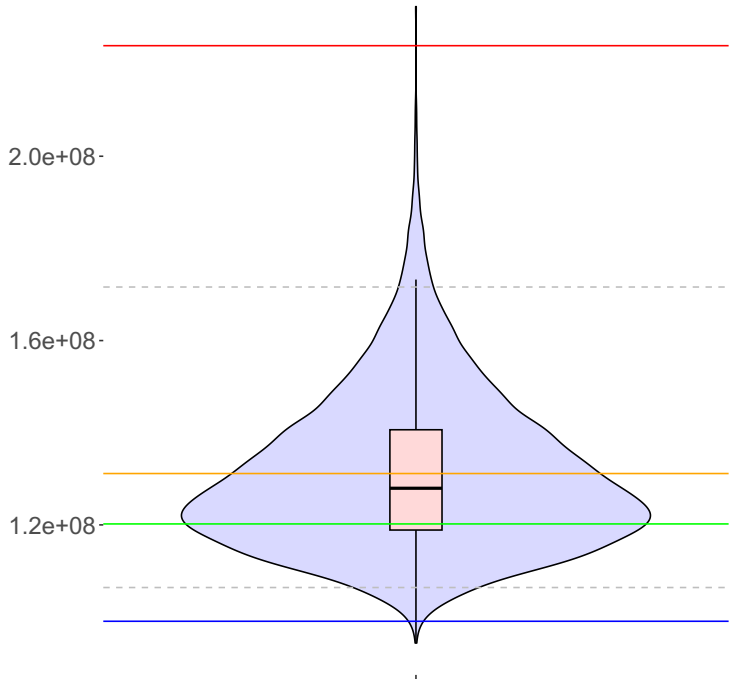


Figure 2.7: The violin plot of the total sum of payments. The blue horizontal line represents the real value of payments, while the green line represents the value of payments calculated via the chain ladder method. The red line represents the estimate of payments calculated via the regression tree method, and the orange line represents the estimate of reserves calculated via the bagging method. The grey dashed lines represent the 95% confidence interval.

We again tried to diagnose the cause of this trouble. In Figure 2.8, we present the estimates for reserves for RBNS claims, and in Figure 2.9 for IBNR claims. We can see that both RBNS and IBNR estimates overestimate the actual values of reserves. We can conclude that the RBNS claims cause the main trouble thanks to the scale.

Table 2.16 displays some main features describing the distribution generated by the bagging approach for both the payments in the next year and the sum of all outstanding payments.

To conclude, both of our methods work pretty well for calculating next year's reserves. However, they overestimate the reserves. The tree method suffers from overfitting while predicting the reserves for $l = 6$, and this makes the total reserves

calculated by this method unreliable. While calculating the reserves for all the upcoming years using the bagging method, we suffer from overestimating. After summing the reserves for years $11, 12, \ldots, 19$, this results in the fact that the sum of all outstanding reserves overestimates the ultimate reserves by quite a lot.

The Solvency Capital Requirement introduced in section 2.3 thus corresponds

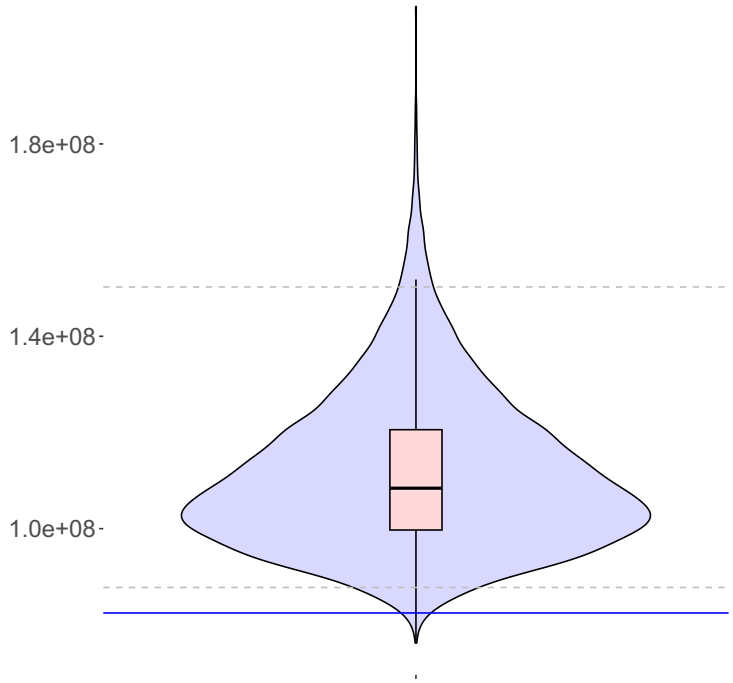


Figure 2.8: The violin plot of the ultimate reserves for RBNS claims. The blue horizontal line represents the real value of payments, while the green line represents the value of payments calculated via the chain ladder method. The red line represents the estimate of payments calculated via the regression tree method, and the orange line represents the estimate of reserves calculated via the bagging method. The grey dashed lines represent the 95% confidence interval.

	next year's	ultimate
real:	44,391,068	99,086,299
mean:	45,609,930	131,146,228
sd:	70,9815	16,920,947
maximum:	48,386,506	232,476,955
99.5% quantile:	47,470,514	188,363,958
97.5% quantile:	47,038,289	171,620,400
75% quantile:	46,083,806	140,646,058
median:	45,599,231	127,951,440
25% quantile:	45,111,226	118,884,735
2.5% quantile:	44,275,804	106,433,734
minimum:	43,024,020	94,354,990

Table 2.16: Summary statistics for next year's and ultimate payments calculated using the bagging technique.

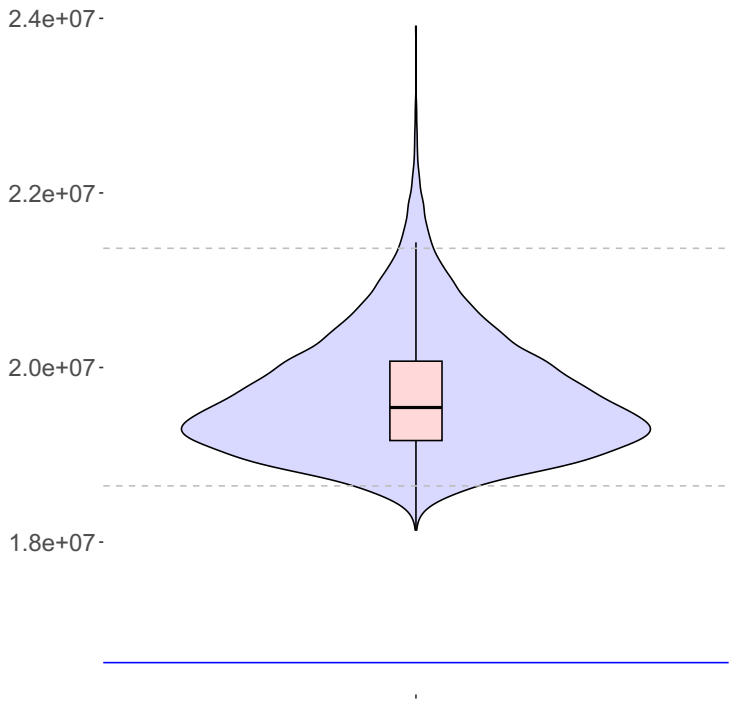


Figure 2.9: The violin plot of the ultimate reserves for IBNR claims. The blue horizontal line represents the real value of payments, while the green line represents the value of payments calculated via the chain ladder method. The red line represents the estimate of payments calculated via the regression tree method and the orange line represents the estimate of reserves calculated via the bagging method. The grey dashed lines represent the 95% confidence interval.

to the 99.5% quantile of next year's reserves and would therefore be estimated by $47,470,514$. The amount of reserves needed for next year, according to the bagging technique, is $45,609,930$ while the actual required reserves were $44,391,068$. The amount of ultimate reserves calculated using the bagging technique is $131,146,228$, while the actual reserves are $99,086,299$.

We can see that both reserves are overestimated. However, the calculation of next year's reserves is much more precise. We can probably attribute this massive overestimation in the second case to the fact that many predicted values were used in further prediction, and thus the errors get magnified.

Conclusion

In this thesis, we introduced classification and regression trees, and bagging. Then we introduced the basic insurance reserving terminology. The main core of the thesis is the application of classification and regression trees and bagging on the reserving problem, i.e., the application of completely different mathematical tools in a practical way on the claim-by-claim reserving problem.

In the theoretical part, we worked on the base of Breiman et al. [1993]. The theory in this book was sometimes not as punctual as it should be. There are several instances in the book when Breiman skipped formal mathematical formulation. These are the instances we tried to put more mathematical formalism into the theory. As our contribution, we could mention the formulation and proof of two theorems regarding the Gini index and cross-entropy and the fact that they fulfill the requirements for an impurity.

The practical part elaborates on the article Wüthrich [2018], and it extends it. We followed Wütrich's suggestions on possible extensions and generalizations in the outline of the article. One of the possible generalizations is to model the claim payment amounts, not only the existence of a claim payment. We did exactly this. Further generalization of the article we did was to implement bootstrap aggregating (bagging), which drastically reduces the variance of the estimates. Moreover, we also introduced the calculations needed for SCR in Solvency II.

The computational part has a lot of work behind it that is not visible as it is involved in the `R` code that we include with the thesis. Another of our contributions was the theoretical derivation of different formulas needed for reserving. Wütrich does not calculate the claim payments on the yearly level as the sum of all payments in a given year. He only calculates the payments for claims with a given accident year paid a given number of years later. However, for instance, for the calculation of SCR, we need to know the outstanding reserves needed for next year regardless of what accident year the claim is from. Thus we needed to invent a new notation and related formulas in order to calculate these quantities.

Overall, the results calculated using the classification and regression trees and bagging are both overestimating the reserves. However, for next year's reserves, the actual reserves are in the 95% confidence interval. We can conclude that we managed to summarize the theory of classification and regression trees together with bagging techniques and added some more mathematical formalism to it. Then we applied this theory to simulated insurance data, and we were able to produce both interval and point predictions about needed reserves.

Bibliography

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. doi: 10.1007/bf00058655.

Leo Breiman, Jerome Friedman, Charles J. Stone, and Rolf A. Olshen. *Classification and Regression Trees*. Chapman & Hall/CRC, Boca Raton, Florida, 1993. ISBN 978-0-412-04841-8.

European Union. Directive 2009/138/EC of the European Parliament and of the Council of 25 November 2009 on the taking-up and pursuit of the business of Insurance and Reinsurance (Solvency II). Official Journal of the European Union, L 335/1, 17 December 2009, 2015. URL `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32009L0138`. [Online; published 25th November 2009; accessed 15th April 2023].

Brandon M. Greenwell. *Tree-based methods for statistical learning in R*. CRC Press, Boca Raton, Florida, 2022. ISBN 978-1-003-08903-2.

Richard J. Verrall and Mario V. Wüthrich. Understanding reporting delay in general insurance. *Risks*, 4(3):25, 2016. doi: 10.3390/risks4030025.

Melantha Wang and Mario V. Wüthrich. Individual claims generator for claims reserving studies: Data Simulation.r. *SSRN Electronic Journal*, 2022. doi: 10.2139/ssrn.4127073.

Mario V. Wüthrich. Machine learning in individual claims reserving. *Scandinavian Actuarial Journal*, 2018(6):465–480, 2018. doi: 10.1080/03461238.2018.1428681.

Mario V. Wüthrich and Michael Merz. Stochastic claims reserving manual: Advances in dynamic modeling. *SSRN Electronic Journal*, 2015. doi: 10.2139/ssrn.2649057.

List of Figures

List of Tables