Data Science in Insurance

Some introductory case studies

Marco Aleandri, PhD, FRM
SA0 Research Dissertation
Institute and Faculty of Actuaries
Email: marco.aleandri@hotmail.it

May 10, 2019

Ai miei genitori

Abstract

In the light of the increasing importance of data in insurance and, as a consequence, in the actuarial practice, this dissertation has a two-fold objective. First of all, it should provide actuaries with an introduction to data science, including basic concepts, terminology, data mining issues, performance measures, machine learning techniques and software. That is necessary as those topics are definitely outside the typical toolkit of actuaries. Secondly, it should suggest significant applications of data science to actuarial problems (e.g., pricing, reserving, ratemaking, etc.). We will look at a number of case studies, investigating the extent to which machine learning can enhance or outperform more traditional approaches.

The dissertation is indeed structured to reach these two goals. While Chapter 1 consists of an introduction to the main concepts of data science, Chapters 2, 3 and 4 describe three different applications in actuarial practice tackled with machine learning techniques. The details on those techniques are outlined just before the applications themselves, in order to keep data science and actuarial practice parallel all the way through the dissertation. To pass on the message that data science can be relevant to any actuary regardless of his/her specific field, the applications involve very different topics. Chapter 2 focuses on marketing and customer behaviour in motor insurance to highlight the importance of data preparation and unsupervised learning. Chapter 3 describes the most common supervised learning techniques as an alternative to regression models in a traditional non-life topic like claim reserving. Finally, Chapter 4 illustrates an example of data science application in life practice, that is, predicting lapse rates to improve asset-liability-management models.

Contents

Chapter 0

Background

The main goal of this project is to represent a systematic introduction to data science for actuaries, leveraging some of those topics that could benefit from it. The dissertation is structured as a data science handbook, but the applications are purely actuarial. Obviously, the dissertation does not account for each and every aspect of data science or actuarial science, but it may be easily enhanced to include new algorithms and examples in the future.
Similar works have been published in the last decade, trying to bridge between daily actuarial practice and more advanced techniques. One of the very first examples is represented by Parodi (2009), where a large number of risk evaluation techniques are introduced, and applied to actuarial topics. To some extent, its structure is similar to that of this dissertation. On the one hand, it is broader and goes beyond data science itself, introducing a large range of computational methods. On the other hand, however, it is limited to general insurance, which lends itself to advanced statistics in a more natural way.
After the aforementioned work, an even higher level of detail is reached in Frees et al. (2014) and Frees et al. (2016), respectively dealing with relevant quantitative techniques in the actuarial field and a variety of applications from several papers. That is the result of a huge effort from many researchers, providing a comprehensive view of quantitative actuarial science in theory and practice. Frees et al. (2014) introduces regression models and other parametric methods, which are then used in Frees et al. (2016). Even if it is not strictly about data science and machine learning algorithms, it has surely inspired this dissertation.
Wüthrich et al. (2018) represents a more recent attempt to connect data science and actuarial practice. It introduces all the main machine learning techniques with a remarkable level of detail, focusing on a variety of motor insurance pricing applications. Once again, the presentation of the methods is extremely comprehensive, but the applications are still limited to a very

specific field, that is, pricing in motor insurance. Of course, that represents a typical actuarial topic where analytics can be successfully applied in its several forms, but it is not the only one.
In the last years, researchers were not the only ones to produce these type of works. Indeed, actuarial associations began to promote data science topics in their working groups, reporting on the opportunities offered by machine learning and big data to the actuarial world.
One of the first contributions was provided by the Belgian actuarial association publishing IABE Information Paper (2015). Although it is quite focused on the business perspective of big data rather than the quantitative aspects of data science, it brings together a lot of interesting ideas on the future of the actuarial profession in the era of data explosion. It covers all the sectors involving actuarial activities, from life to non-life, from pricing to reserving. Emphasizing business-related problems such as customer management, claim reserving and policyholder behaviour, it has somehow suggested the three applications we will present. At the same time, in spite of its lack of quantitative analysis, it has inspired the conclusion to this dissertation in the last chapter.
Another, more recent contribution is represented by IFoA (2018) published by the Modelling, Analytics and Insights in Data working party of the Institute and Faculty of Actuaries. Even if it is much briefer than this dissertation, it shares some of its goals, that is, introducing data science to actuaries and applying machine learning to actuarial case studies. While the former is reached by handling most of the concepts handled in this dissertation as well, the latter is reached through different applications. In particular, IFoA (2018) uses supervised learning for interest rate prediction, marine hull pricing, catastrophe exposure management, and suicide rate estimations.
In IFoA (2018), the business perspective is still present and sometimes predominant, but the discussion encompasses many fundamental data science ideas. Part of the concepts expressed in this dissertation are introduced there as well, although from a high-level perspective and without the necessary details to understand what is really behind each algorithm. Even if the applications do not represent the heart of that work, they touch various actuarial sectors just like we will do here, aiming to demonstrate the potential of alternative techniques against traditional methods. Those aspects make IFoA (2018) different to all the aforementioned works, raising interest among actuaries from any background and sector.
In this dissertation, we aim to pick the best features from the cited works, in order to build a comprehensive, detailed and actuary-oriented introduction to data science. It is essentially structured to answer the following questions:

- *What?* That is: data, dataset partitions, performance measures, software, supervised learning, unsupervised learning, algorithms, etc. Some of those topics will be first presented in Chapter 1, especially

concepts representing the foundations of data science, which will be then useful throughout the dissertation. By contrast, the peculiarities of algorithms will be covered in detail as soon as we will need them for the actuarial applications of the central chapters, in order to better connect machine learning technicalities and specific case studies. Considerable effort will be devoted to maintain the typical structure of data science manuals in terms of sequence of the topics.

- *How?* That is: actuarial applications for different sectors of the actuarial practice. More specifically, we will tackle three case studies: customer management in Chapter 2, individual reserving in Chapter 3 and policyholder behaviour in Chapter 4. They will be respectively handled by using unsupervised learning, supervised learning and ensembles (i.e., combinations of different methods). Data availability is not the only reason why we choose those topics. First, we want to tackle things that are potentially part of the daily actuarial practice such as renewal rates, claim reserves or lapse probabilities. Second, we want to tackle things involving different actuarial areas such as underwriting, non-life business and life business. These two targets aim to raise the interest of actuaries regardless of their specific background and role in the industry.

- *Why?* That is: accuracy or inaccuracy, stability or instability, interpretability or black-box effect, etc. Highlighting relevant reasons to prefer data science over traditional approaches is crucial. As a first step, Chapter 3 will demonstrate the importance of unsupervised learning in data manipulation as well as its potential in detecting clusters and improving accuracy. However, this will come at the cost of increase in model complexity. Instead, Chapter 2 will show that more flexible machine learning techniques such as decision trees may outperform regression models. Even if this is not a statement that holds in general, it will provide actuaries with suitable alternative methods to boost model performance. Actually, Chapter 4 will start illustrating that those alternatives may miserably fail because of instability due to data flaws, algorithmic issues or other problems. That will justify the usage of ensembles: in particular, bagging trees will imply more stability and outperformance over logistic regression.

This is just a first step to make data science methods relevant to the whole actuarial world, in both academia and industry.

Chapter 1

Data Science Basics

As suggested by the title, this first chapter will represent a brief introduction to the main concepts used in data science. They encompass general and soft notions (e.g., classification and prediction, supervised learning and unsupervised learning, etc.) as well as more precise and technical definitions (e.g., bias and variance, performance measures, etc.).

Chapter overview

The first part will provide the reader with an overview of the current role of data science in business, describe the data mining process through its main standards, and highlight the importance of data availability. Subsequently, the second part will introduce some typical performance evaluation tools and other criteria for model selection, define generalized linear models that are relevant to actuarial practice, and list the most common machine learning techniques.
Most of those concepts will be used throughout the dissertation on several occasions, although they are not entirely part of the traditional actuarial background. Therefore, the next pages aim to build the necessary foundation for the following chapters.

1.1 Some terminology

When it comes with statistical methods, there are quite a lot of terms that are spreading among actuaries nowadays. Nonetheless, actuaries are not really statistical experts since their education is focused on statistics to the extent they are effectively applicable to insurance. This first section aims to introduce some basic concepts that will turn out to be useful throughout the dissertation. However, it is not meant to be fully comprehensive, and it can easily get out of date as new concepts come out. For our aims, however, it should be enough.

The very first concept to introduce is that related to *data science*. It denotes the scientific field about the entire range of systems, processes and methods used for *data mining*, that is,

> *the process of exploration and analysis, by automatic or semi-automatic means, of large quantities of data in order to discover meaningful patterns and rules*

as defined in Berry et al. (1997). Therefore, data science is much more than statistics. Among others, it covers data integration, data architecture, data visualization, data engineering, data-driven business analysis and of course the whole range of tools to mine data.
Some of these tools come from classical statistics, for instance, sampling methods, confidence intervals, hypothesis tests and regression, just to mention the major ones. To some extent, all of them are based on analytical assumptions and mathematical formalization. It guarantees a strong, theoretical foundation to these tools, so that they may be used in any relevant application.
Other tools lack such a theoretical strength, but gain much more flexibility to recognize pattern in data. More specifically, they are structured to automatically adapt their input parameters in order to catch more and more information from a given dataset. This is the reason why it is often said that such tools "learn" from data. The statistical field that encompasses all of them is called *machine learning*, the object of this dissertation.
In machine learning, two types of "learning" are usually mentioned:

- *supervised learning*, that is, learning about the relations between a range of *predictors* (so-called *explanatory variables* in regression) and a determined *target* (so-called *response variable* in regression);
- *unsupervised learning*, that is, learning about the relations between a range of variables in order to group "similar" records.

In both the cases, the algorithm catches information, and uses it to interpret new data. In supervised learning, it will use new data predictors to predict the related target variable. In unsupervised learning, it will use new data variables to assign new records in previously identified clusters. Given a dataset, one may use supervised tools or unsupervised tools depending on the goal - prediction for the former, segmentation for the latter - but there is no difference in the data. Just remember that, in supervised learning, we distinguish variables between a range of predictors and one single target, while there is not such a distinction in unsupervised learning.
Nonetheless, a further distinction is relevant in supervised learning, depending on the nature of the target variable. A specific tool is used for

- *classification* if the target variable is categorical

- *prediction* if the target variable is numerical
- *forecasting* if the target variable is a time series.

The great majority of machine learning tools can be adapted for both classification and prediction, while the tools for forecasting time series are usually considered in isolation. In our analyses, we will only consider tools for classification and prediction such as decision trees and neural networks. The map in Figure 1.1 shows the most common ones (sometimes, there is a lack of uniformity in naming different tools among researchers, so one could find various names for essentially the same tool). However, on a daily basis, brand new algorithms or improvements to old tools are created by data scientists to tackle specific problems, especially in the last years. Rather than explaining all of them, we are going to focus on the most widely used.
Quite interesting is the relationship between machine learning and *artificial intelligence*. The two concepts share something similar and tend to be confused with each other, but are quite different in reality. Artificial intelligence was born in the 1960s as a subfield of computer science with the main goal of programming computers to perform human tasks (e.g., speaking, listening, writing, translating and many more). Actually, some of them are so complex that artificial intelligence needs specific machine learning tools. However, the same tools could just be as successful in any other field, say, predicting stock prices. At the same time, one might use non-learning tools in artificial intelligence, if they are sufficient to replicate some human behaviours (for example, imagine a very complex algorithm with some fixed parameters: it will run always the same way, without learning anything from new data). In reality, human beings are so complex that machine learning tools are considered a "must" in artificial intelligence. Contemporarily, artificial intelligence is seen as a main goal for machine learning. This is the reason why they get often confused.
Nonetheless, artificial intelligence is not the only field where machine learning techniques have been proven to be useful. A less "noble", but more practical field is *business intelligence*. Generally, the concept of business intelligence comprises all the data processes, data technologies and data insights aimed to the improvement of specific business activities and performances. Forrester.com reports the following definition:

> *Business Intelligence is a set of methodologies, processes, architectures, and technologies that transform raw data into meaningful and useful information used to enable more effective strategic, tactical, and operational insights and decision-making.*

If we compare this definition to that of data mining, it will be clear that business intelligence is just data mining from a business perspective, in an industry environment. As such, it also focuses on the practical aspect of

Fig. 1.1: A map of machine learning techniques (see Brownlee (2013))

data mining, for instance, data visualization (e.g., reporting tools and executive dashboards), decision-making process and so on. While data mining and machine learning are rather standardized concepts, business intelligence features are quite customizable by industry. For example, a specific industry professionals usually prefer certain machine learning tools or software, because they have been already proven to be effective in that field. Here is a list of actual, specific and successful examples of business intelligence:

- E-commerce companies need machine learning to draw data insights about a number of daily issues, for instance, which products would be preferred by which customers, which customers would go for which promotions, which products would be purchased together or in a short time-frame and so on (this is an example of *market basket analysis*, introduced later in the dissertation);
- Social network developers use machine learning to analyse users emotions after status updates (this is an example of *sentiment analysis*);
- Search engines such as Google and Yahoo cluster "similar" web pages by using unsupervised learning techniques;
- E-mail spam filtering always rely on some simple machine learning tools to detect spam;
- Financial institutions use machine learning to forecast the stock market, and thus direct investment decisions;

- Banks classify loan and mortgage applicants by different level of riskiness using the default probability predicted by some machine learning tools;
- Healthcare industry relies on machine learning in several applications, for instance, detecting patients who will likely develop a chronic disease, and predicting possible adverse drug reactions in patients;
- Transport companies use machine learning to forecast customer behaviour and thus real needing in transportation in order to reduce costs;
- Automobile industry companies predict the failure or breakdown of mechanical parts by using machine learning;
- Speech, image and text recognition are some of the most recent - and successful - applications of machine learning in the field of information technology.

This list is very far to be complete: actual machine learning applications can be found in countless fields. But what should really surprise us is that it is only the beginning. Data science has the potential to improve (e.g., increasing quality, reducing costs, etc.) each and every human activity. In some cases, such an improvement will turn into disruption, so that the foundations themselves of some fields will adapt to the new technologies. To some extent, it could be similar to the recent digital revolution in terms of capacity.
Of course, actuaries should align throughout the world, and this dissertation is especially devoted to them. After the brief introduction to machine learning in this chapter, we will present some practical, actuarial problems that could be actually solved by new techniques. It will give the reader a fair idea of their potential, and maybe he/she will find out other daily issues that could be solved in the same way.

1.2 Standards for the data mining process

As we have just discussed, machine learning is only a little component of the data mining process. At the same time, business intelligence usually takes a different form depending on the industry it is applied; thus the data mining process itself is quite prone to specific changes. Consequently, in the last two decades, the needs for a standard data mining process increased dramatically, so several data mining process models have been proposed. According to Piatetsky (2002), Piatetsky (2004), Piatetsky (2007) and Piatetsky (2014), the most widely recognized as a standard framework is the *Cross Industry Standard Process for Data Mining* (CRISP-DM). On the

initiative of SPSS, Teradata, Daimler AG, NCR Corporation and OHRA, the project started in 1996. Four years later, Chapman et al. (2000) was published, a first guide on the data mining process.
CRISP-DM breaks the data mining process into the following six major phases:

C1 *Business understanding*: identifying relevant business goals for the data mining process, and structuring a project plan to design it in order to meet them;

C2 *Data understanding*: collecting data, assessing data quality, detecting some information through basic descriptive statistics or data visualization, and drawing the first conclusions;

C3 *Data preparation*: preprocessing raw data through various preliminary activities (e.g., data reduction, clustering, variable selection, variable transformation, handling missing data, outliers and categorical variables and so on) in order to build the final dataset;

C4 *Modelling*: selecting a range of machine learning tools, setting them to suit the dataset, and applying them to the business problem;

C5 *Evaluation*: collecting predictive performances from the different tools, comparing them to pick the best model, and review the process to identify issues and weaknesses;

C6 *Deployment*: communicating insights to the management, providing documentation and executive reports, monitoring assumptions and model, updating data on a regular basis and any other maintenance activity.

The six phases and the main features of each phase are also illustrated in Figure 1.2 and 1.3. In particular, notice that the sequence of the phases is not meant to be strict, as illustrated by the arrows in Figure 1.2. For instance, while building some algorithms in the modelling phase, it may happen that we need further data transformation to adapt the dataset to the actual algorithm. In such a case, we will get back to the data preparation phase to perform the necessary tasks, and then to the modelling phase once again.
Although CRISP-DM is surely the most accepted standard, others are often cited. In particular, the *Knowledge Discovery in Databases* (KDD) deserves consideration as the ancestor of CRISP-DM. It was developed in 1996 and described in a series of sources (see Fayyad et al. in MIT Press (1996), Fayyad et al. in AI Magazine (1996) and Fayyad et al. in KDD-96 (1996)). The KDD process comprises five phases:

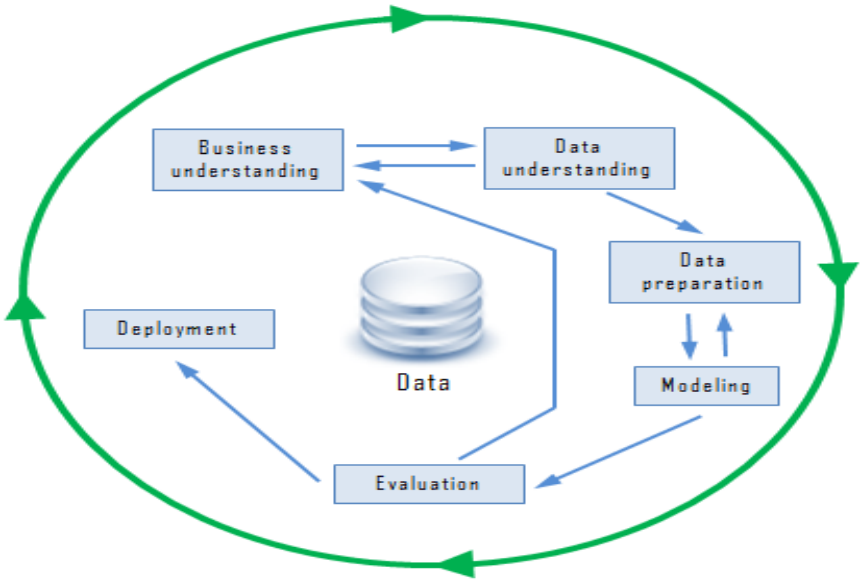


Fig. 1.2: CRISP-DM reference model (see Chapman et al. (2000))

Business Understanding	Data Understanding	Data Preparation	Modeling	Evaluation	Deployment
Determine Business Objectives *Background* *Business Objectives* *Business Success Criteria* **Assess Situation** *Inventory of Resources* *Requirements, Assumptions, and Constraints* *Risks and Contingencies* *Terminology* *Costs and Benefits* **Determine Data Mining Goals** *Data Mining Goals* *Data Mining Success Criteria* **Produce Project Plan** *Project Plan* *Initial Assessment of Tools and Techniques*	**Collect Initial Data** *Initial Data Collection Report* **Describe Data** *Data Description Report* **Explore Data** *Data Exploration Report* **Verify Data Quality** *Data Quality Report*	**Select Data** *Rationale for Inclusion/ Exclusion* **Clean Data** *Data Cleaning Report* **Construct Data** *Derived Attributes* *Generated Records* **Integrate Data** *Merged Data* **Format Data** *Reformatted Data* *Dataset* *Dataset Description*	**Select Modeling Techniques** *Modeling Technique* *Modeling Assumptions* **Generate Test Design** *Test Design* **Build Model** *Parameter Settings* *Models* *Model Descriptions* **Assess Model** *Model Assessment* *Revised Parameter Settings*	**Evaluate Results** *Assessment of Data Mining Results w.r.t. Business Success Criteria* *Approved Models* **Review Process** *Review of Process* **Determine Next Steps** *List of Possible Actions* *Decision*	**Plan Deployment** *Deployment Plan* **Plan Monitoring and Maintenance** *Monitoring and Maintenance Plan* **Produce Final Report** *Final Report* *Final Presentation* **Review Project** *Experience Documentation*

Fig. 1.3: Generic tasks (bold) and outputs (italic) of the CRISP-DM reference model (see Chapman et al. (2000))

K1 *Selection*: selecting relevant variables from the dataset, in accordance with the object of the analysis;

K2 *Preprocessing*: cleaning raw data to build the final dataset;

K3 *Transformation*: reducing data and transforming variables to make it suitable for data mining;

K4 *Data Mining*: setting and using various machine learning tools to discover patterns in data in line with the object of the analysis;

K5 *Interpretation/Evaluation*: comparing predictive performances from different tools in order to pick the best model, and interpreting the results.

Another significant contribution to the standardization of the data mining process came from SAS Institute in 2008. Its framework is called SEMMA, from the initials of the five phases which it consists of:

S1 *Sample*: data sampling and partitioning;

S2 *Explore*: data visualization;

S3 *Modify*: variable selection and transformation;

S4 *Model*: application of various machine learning tools to provide the desired outcome;

S5 *Assess*: comparison of different predictive performances from different tools.

SAS Institute points out that SEMMA is not meant to design a generic data mining methodology, rather it is just an internally developed framework representing the foundation of their services. It is especially the case of their data mining software, SAS Enterprise Miner, whose the main five tabs are labelled by the names of the five SEMMA phases. Nonetheless, Piatetsky (2002), Piatetsky (2004), Piatetsky (2007) and Piatetsky (2014) reveal that SEMMA is often used as a general guidance for data mining in industry - second only to CRISP-DM.
In Figure 1.4, we report a formal comparison between the phases of the three aforementioned standards (see Azevedo et al. (2008)). Some of them overlap with each other, but the table gives a fair indication of the reason why CRISP-DM is the most suitable to be considered as the best practice worldwide. Indeed, it is the only one including the business intelligence perspective in data mining, at the beginning (business understanding) as well as at the end (deployment) of the process.
After each of our applications (in the final sections), we will explicitly refer to all the CRISP-DM phases, except for the deployment, given that it

CRISP-DM	KDD	SEMMA
Business understanding	-	-
Data understanding	Selection	Sample
Data understanding	Preprocessing	Explore
Data preparation	Transformation	Modify
Modelling	Data Mining	Model
Evaluation	Interpretation/Evaluation	Assess
Deployment	-	-

Fig. 1.4: Correspondences across CRISP-DM, KDD and SEMMA

is more related to the model management at enterprise level. Nonetheless, since our scope is not purely technical, we need to go beyond data and data mining. Finding a machine learning tool that provides us with valuable insights is just a part of the work. Indeed, the ultimate question is: how does it apply to the actuarial practice, and which is the impact on the insurance business?

1.3 The role of data

Before going through any further detail about data science tools and applications, it should be clear to the reader that all those concepts are practically useless without a reasonable amount of significant data. That is true in classical statistics too, but "less fundamental", in the sense that it only requires data to fulfil some assumptions, while the existence itself of machine learning actually relies on data.
Of course, the first problem is about gathering data. The actuarial applications that will be presented later were possible thanks to data availability. Moreover, their outcomes may be slightly - if not even fundamentally - different if data changes. In other words, those applications are supposed to provide actuaries with broad ideas to apply data science in their own field, but they are not supposed to draw general conclusions on the related phenomenons. Our analyses will be only empirical as the related datasets are unique. Nevertheless, where possible, we will try to compare our outcomes with the results of similar studies, in order to have an indication about their validity.
If data is actually there, then the next question to pose is: is it reliable as it is? Often, this is not the case, in the sense that it would need a considerable amount of data preparation and manipulation before machine learning could actually step in. As we will see, our applications will also need that

in several forms. In this phase, the most basic tasks include simple operations on data, for instance, various transformations such as log-conversion or standardization, missing data handling, field selection, record filtering, binarization of categorical variables, categorization of numerical variables and so on. Sometimes, we could even need more complex techniques such as clustering. Of course, each of those tasks need to be adapted to the specific case, and it is not always clear how. Except for (very) few, common-sense rules, one can hardly generalize data preparation. In data science, this is a fundamental step though, so we will always dedicate part of our actuarial applications to pure data preparation.
A last data issue regards the infrastructure. Even if data is there, and has been properly prepared for further analysis, our infrastructure could be inappropriate to handle it. This is not only due to data size (obviously, big data requires more and more complex resources than small data), but also data form. Structured data such as numbers and characters are relatively easy to handle, but what about unstructured data such as e-mails, audio files, photos and so on? Handling them requires much more effort and knowledge. Although it could sound far from the tasks that actuaries face on a daily basis (and actually these are not issues that we will encounter throughout the dissertation), this type of data may increasingly represent useful areas to explore as data science starts playing an important role in actuarial practice.

1.4 Performance evaluation

Essentially, regression models are the preferred tools among actuaries for two reasons. First, they can be easily communicated - a target variable is just a regular function of various predictors, and the dependency is nicely measured by a parameter. Second, they can be easily defended - under specific assumptions, regression models return the best possible estimation. That's always true, regardless of how bad the results are.
As soon as such a theoretical framework is removed, we lose all the quantitative measures related to it, e.g. confidence intervals, R^2, p-values and so on. Unfortunately, machine learning does not provide us with comparable tools, that is, we need to find something else to evaluate model performance.

1.4.1 Training, validation, test

In supervised learning, we first get help from the data mining process itself. In the data preparation phase, indeed, the entire dataset is usually partitioned into three parts:

- *training dataset*: it is used to build the model(s) we want to apply to our specific classification or prediction problem;

- *validation dataset*: it is used to evaluate the performance of the model(s) we have just trained;
- *test dataset* (optional): it is meant to consist of some external data, so it is used to provide an ultimate performance evaluation of the model(s).

This partition represents the typical first step in data mining, as explained in most of the data science sources (for instance, see Hastie et al. (2009) and Shmueli et al. (2010)).
The training dataset is the most important one: if it is irrelevant for some reasons, the model(s) will be somehow incomplete, and the related performance evaluations from the validation data and the test data will be flawed. Thus, it is usually the largest partition, in order to contain all the relevant information. Once the training records have been selected and picked out, the whole remaining dataset may be considered as the validation partition. In other words, we do not necessarily need a test dataset if there are no relevant features that differentiate test data from validation data. For instance, we will use a test dataset for the case reserving application (see Chapter 3), but no test dataset will be considered for the policyholder behaviour application (see Chapter 4). All in all, the choice mainly depends on the specific dataset.
As we said, the training dataset will build our model, that is, it will try to "learn" as much as possible from the training records. For this reason, such a model will be especially able to detect information from those records. But what if we run the model on data which it did not learn from? Sometimes, machine learning algorithms seem very powerful on training data, but show weaknesses in predicting new data such as validation data. Generally, as the complexity of an algorithm increases to catch all the information in the training data, the algorithm starts *overfitting* the validation data. Since it fully reflects the training dataset without distinguishing between "signal" and "noise", the noisy component cause greater error in the validation dataset. In fact, the typical consequence of overfitting is that, after some unknown level of complexity, the prediction error on the validation dataset stops decreasing and starts increasing (see Figure 1.5). As a consequence, two questions are crucial when we need to evaluate the performance of a given algorithm:

Q1 *is the training performance similar enough to the validation performance? If not, is it due to overfitting?*

Q2 *is the validation performance good enough for our purposes? Is it better than that of other algorithms?*

While the answer to Q1 provides us with an indication about the actual predictive power of the model, the answer to Q2 allow us to make our

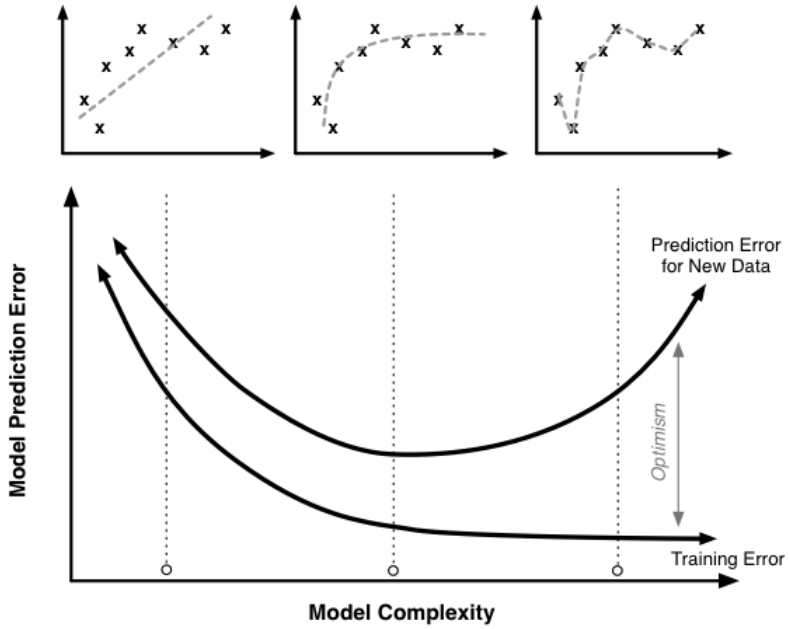


Fig. 1.5: Training error versus validation error by model complexity (see Nogueira (2016))

choice among a number of models - and it will be based on the validation performance only. Actually, when it comes with model selection we are barely interested in the training performance.

Dataset partitioning is a basic characteristic of the data mining process. As such, it is applied regardless of the specific machine learning tools used, including regression models. However, there are two reasons why this is much less relevant in regression theory.

First of all, regression models are calibrated to guarantee the best *fitting* rather than the best prediction. Although machine learning tools are trained on the training data only, their complexity features are set to return the lowest validation error. For instance, Figure 1.5 suggests to choose the model with complexity corresponding to the vertical dashed line in the middle, but the algorithm is still trained on the training data. On the contrary, a regression model is entirely based on a single (training) dataset to fit it at best, so there is no reason why it should also result in good predictions on new data.

Additionally, regression models are rigidly defined on analytical assumptions such as target variable distributions and link functions. Parameters are thus calibrated based on those assumptions, rather than actual training from data. This feature gives them a unique tendency to ignore noise in

data. This is the reason why overfitting is seldom a problem in regression, especially if explanatory variables have been preliminarily selected with a proper selection algorithm.

1.4.2 Performance measures

In Subsection 1.4.1, we dealt with concepts such as training error and validation error. They are rather abstract concepts as long as we don't define them analytically. For instance, regression theory provides us with a natural definition of error consistent with the underlying theory, the so-called *sum of the squared errors*:

$$SSE := \sum_{i=1}^{N} u_i^2 := \sum_{i=1}^{N} (y_i - \widehat{y}_i)^2 = \sum_{i=1}^{N} \left[y_i - \widehat{f}(\boldsymbol{x}_i, \widehat{\boldsymbol{\beta}}) \right]^2 \tag{1.1}$$

where N is the total number of records, and u_i denotes the estimation error defined as the difference between the actual value y_i of the target variable and the predicted value $\widehat{y}_i$, which results from a predefined estimation function $\widehat{f}$ of the feature vector $\boldsymbol{x}_i$ and the estimated parameter vector $\widehat{\boldsymbol{\beta}}$ (notice that it is exactly the error measure minimized in Subsection 1.6.1 to obtain the parameter estimations $\widehat{\beta}_0, \ldots, \widehat{\beta}_n$). By contrast, machine learning tools do not rely on a common theory, so we get more flexibility in defining the error measure in machine learning. In this section, we will describe the most common measures in classification and prediction.
In classification problems, we want to estimate the probability of different outcomes, that is, the possible classes of the target variable. Given the k^{th} class and the i^{th} record, if p_{ik} and q_{ik} denote the related actual frequency (which equals 1 if the i^{th} record belongs to the k^{th} class, otherwise 0) and estimated probability respectively, then the likelihood of the dataset is

$$L(\boldsymbol{p}, \boldsymbol{q}) := \prod_{i=1}^{N} \prod_{k=1}^{m} q_{ik}^{p_{ik}} \tag{1.2}$$

which implies the following log-likelihood:

$$l(\boldsymbol{p}, \boldsymbol{q}) := \ln L(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{N} \sum_{k=1}^{m} p_{ik} \ln q_{ik} = \sum_{i=1}^{N} \ln q_{ik_i} \tag{1.3}$$

where k_i denotes the actual class of the i^{th} record. Maximizing the log-likelihood (1.3) is equivalent to minimizing the so-called *cross entropy*

$$H(\boldsymbol{q}) := -\sum_{i=1}^{N} \ln q_{ik_i} \tag{1.4}$$

Actual Class	*Predicted Class* 1	0
1	TP	FN
0	FP	TN

Fig. 1.6: Classification matrix for binary target variables

Actual Class	*Predicted Class* 1	2	...	$m-1$	m
1	T_1	$F_{1,2}$	...	$F_{1,m-1}$	$F_{1,m}$
2	$F_{2,1}$	T_2	$\ddots$	$F_{2,m-1}$	$F_{2,m}$
$\vdots$	$\vdots$	$\ddots$	$\ddots$	$\ddots$	$\vdots$
$m-1$	$F_{m-1,1}$	$F_{m-1,2}$	$\ddots$	T_{m-1}	$F_{m-1,m}$
m	$F_{m,1}$	$F_{m,2}$	...	$F_{m,m-1}$	T_m

Fig. 1.7: Classification matrix for categorical target variables

which is indeed the main performance measure in classification, although not the only one.
In fact, the simplest tool for performance evaluation in classification problems is the *classification matrix*. Figure 1.6 illustrates a generic classification matrix for a binary target variable, while Figure 1.7 illustrates a generic classification matrix for a categorical target variable. It gives a quick overview of how many records are correctly or incorrectly classified. Indeed, in Figure 1.7, T_k denotes the number of actual records in the k^{th} class that are correctly classified in that class by the algorithm we are using. By contrast, $F_{k,h}$ denotes the number of actual records in the k^{th} class that are incorrectly classified in the h^{th} class by the same algorithm. As shown in Figure 1.6, they are often called *true positives* (TP), *true negatives* (TN), *false positives* (FP) and *false negatives* (FN) in binary classification. Finally, a main error measure in classification is the *misclassification error*:

$$ME_m := \frac{\sum_{i,j:i\neq j} F_{i,j}}{N} = \frac{\sum_{i,j:i\neq j} F_{i,j}}{\sum_{i,j:i\neq j} F_{i,j} + \sum_{i=j} T_i} \tag{1.5}$$

which, in binary classification, equals

$$ME_2 = \frac{F_{0,1} + F_{1,0}}{N} = \frac{F_{0,1} + F_{1,0}}{F_{0,1} + F_{1,0} + T_0 + T_1} = \frac{FP + FN}{FP + FN + TN + TP}. \tag{1.6}$$

Even if the misclassification error is an overall measure for the performance of an algorithm, there are others, class-specific measures. First, we may be

interested in the misclassification error related to the generic class k:

$$ME_m(k) := \frac{\sum_{j \neq k} F_{k,j}}{\sum_{j \neq k} F_{k,j} + T_k}, \quad \forall k = 1, \ldots, m \tag{1.7}$$

which, in binary classification, equals

$$ME_2(1) = \frac{F_{1,0}}{F_{1,0} + T_1} = \frac{FN}{FN + TP}, \quad ME_2(0) = \frac{F_{0,1}}{F_{0,1} + T_0} = \frac{FP}{FP + TN}. \tag{1.8}$$

Parallel to error measures, we may consider accuracy measures as well. For instance,

$$A_m := 1 - ME_m = \frac{\sum_{i=j} T_i}{\sum_{i,j:i \neq j} F_{i,j} + \sum_{i=j} T_i} \tag{1.9}$$

is a natural definition for the overall accuracy of a classification algorithm, and its binary version is

$$A_2 = 1 - ME_2 = \frac{TN + TP}{FP + FN + TN + TP}. \tag{1.10}$$

Also, the accuracy related to the generic class k is defined as

$$A_m(k) := 1 - ME_m(k) = \frac{T_k}{\sum_{j \neq k} F_{k,j} + T_k}, \quad \forall k = 1, \ldots, m \tag{1.11}$$

and, in binary classification,

$$A_2(1) = 1 - ME_2(1) = \frac{TP}{FN + TP}, \quad A_2(0) = 1 - ME_2(0) = \frac{TN}{FP + TN} \tag{1.12}$$

which are called *sensitivity* and *specificity* respectively, and are two very common measure in binary classification. Actually, when a target variable is binary, one of the two categories is generally much more relevant than the other one. This is also the reason why the class 1 is often referred to as "success", while the class 0 represents a "failure". Whatever the application, we are typically more concerned with the correct classification in class 1. From our model, we ideally want to get

- low $ME_2(0)$, namely the complement of the specificity, that is, few false positives
- high $A_2(1)$, namely the sensitivity, that is, numerous true positives

but here is a trade-off to consider. In effect, assume that 0,5 is the *cut-off value*, that is, if the estimated probability of a record to be an actual "success" is higher than 0,5, then it will be classified in class 1, otherwise it will be classified in class 0. This choice will imply specific $ME_2(0)$ and

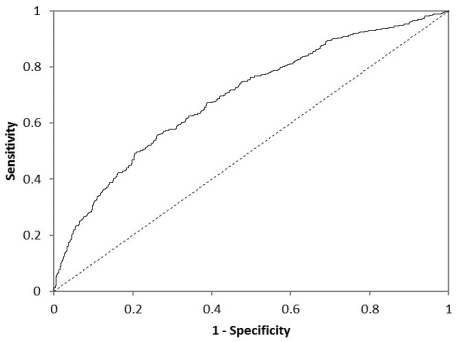


Fig. 1.8: Example of ROC curve

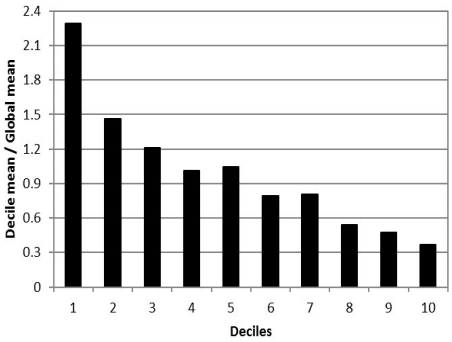


Fig. 1.9: Example of decile chart

$A_2(1)$. Now, assume we want to increase the sensitivity, in order to better classify successes: for that purpose, we may decide to set the cut-off to 0,3. This would surely increase the sensitivity since the records classified in class 1 would be more than before, but at the same time it would also worsen classification in class 0, that is, increase in $ME_2(0)$. All in all, we may want to try various cut-off values until we find one that balances $ME_2(0)$ and $A_2(1)$.

More often, however, these two measures are used to plot the *Receiver Operating Characteristic curve*, or ROC curve. It plots $ME_2(0)$ on the x-axis and $A_2(1)$ on the y-axis varying by cut-off value from 0 to 1. An example is shown in Figure 1.8. Better performance is reflected by curves that are closer to the top left corner, that is, with an *area under the curve* (AUC) as close as possible to 100%. In particular, the diagonal dashed curve in Figure 1.8 represents the random model, that is, the model that estimates the probability of success as the overall proportion of actual successes in the dataset, regardless of the specific record (*naïve rule*). Its AUC is 50%, the smallest possible. In theory, no classification algorithm should perform worse than the random model, so the ROC curve will always lay above the diagonal line in Figure 1.8, and the AUC will always lay between 50% and 100%.

The *decile chart* in Figure 1.9 is another useful tool for performance evaluation. When a classifier returns a probability of "success" and "failure", the records can be sorted by decreasing estimated probability of "success". In terms of classification, the list separates records classified as "successes" and records classified as "failures". If the classifier performs well, we should also get most of the *actual* "successes" at the top, while most of the *actual* "failures" stay at the bottom.

To evaluate the performance of the classifier, we consider the ten deciles - the quantiles representing 10% of the (sorted) dataset - and count the number of true positive per decile. Since the records are sorted by estimated probability, we expect true positives especially in the first deciles. For sake

of simplicity, assume that in the first decile we get predicted "successes" only (although it is quite common in practice, it may not occur if "successes" are very rare or the model is very poor). Our classifier returns the proportion p_1 of true positives in the first decile. Such a proportion may be compared to the proportion we would have got from the random model, that is, the empirical proportion s of "successes" in the dataset. The ratio $\frac{p_1}{s}$ provides an indication about the performance of the model in predicting "successes" in the first decile of the distribution. Naturally, we can do the same with the others deciles, computing the ten ratios $\frac{p_1}{s}, \ldots, \frac{p_{10}}{s}$, which correspond to the bars of the decile chart in Figure 1.9. For instance, the first bar indicates a ratio around 2,3: the classifier performs 2,3 times better than the random model. Since classifiers are generally constructed to predict "successes" rather than "failures", the typical shape of a decile chart decreases across the deciles. The classifier is more powerful in the top (more interesting) deciles, while it fails in classifying "successes" hidden in some bottom decile. This is the reason why we pay much more attention to the first bars, whose ratios should be significantly greater than 1.
Notice that decile charts are useful for prediction of numerical variables too. In this case, each bar is based on the ratio of the related decile mean to the mean over the entire dataset. However, there is no equivalent of the ROC curve or misclassification error. In numerical prediction, we can use a range of accuracy measures that are function of the residuals $u_i := y_i - \widehat{y}_i$. All of them are analogue to the traditional sum of squared errors in (1.1), and include the followings:

$$\text{mean error} := \frac{1}{N}\sum_{i=1}^{N} u_i \tag{1.13}$$

$$\text{mean absolute error} := \frac{1}{N}\sum_{i=1}^{N} |u_i| \tag{1.14}$$

$$\text{mean relative error} := \frac{1}{N}\sum_{i=1}^{N} \left|\frac{u_i}{y_i}\right| \tag{1.15}$$

$$\text{mean squared error} := \frac{1}{N}\sum_{i=1}^{N} u_i^2 = \frac{SSE}{N} \tag{1.16}$$

$$\text{root mean squared error} := \sqrt{\frac{1}{N}\sum_{i=1}^{N} u_i^2} = \sqrt{\frac{SSE}{N}}. \tag{1.17}$$

as reported in Shmueli et al. (2010). Mean absolute error and mean squared error (or equivalently root mean squared error) are the most widely used in training algorithms and evaluating predictive performances. The former (as well as the mean relative error) is indeed the most intuitive way to

define the overall estimation error, while the latter (as well as the root mean squared error) is supported by the regression theory as it is analytically advantageous. By contrast, the mean error is seldom used, because it will net errors that have opposite sign.
Finally, it is worth mentioning the cost adjustment that may be embedded in performance measures, although we will not use them in our analyses. As we have already discussed, the goal of a model is the prediction of "successes" rather than "failures". This does not mean that we are not interested in false positives, but it means that we are prone to accept more of them if it leads to significantly fewer false negatives. In classification, for instance, we can express that by adjusting the cut-off manually. It is 0,5 by default, but if we set it to 0,3 then we are implicitly assuming that we accept more false positives in order to reduce false negatives. If an insurance company is selling a new product, and we built a model that classify its customers as "interested" and "uninterested", the company will surely accept the risk of contacting a higher number of potentially uninterested customers if it materially increases the probability of capturing the interested ones. This is due to the cost of the contact: if the contact is represented by a simple mail, the cost is quite low, so the company is not really worried about false positive, but it knows that each false negative costs the entire premium of the product.
While the cut-off adjustment is heuristic and manual, the cost adjustment on the performance measure is based on the estimated cost of an error. In the previous example, the cost of false positive is the cost of a mail, whereas the cost of a false negative is the missed premium. If these quantities are known - say c_{FP} and c_{FN} - we define the *misclassification cost*

$$MC_2(c_{FP}, c_{FN}) := \frac{c_{FP}FP + c_{FN}FN}{N} \tag{1.18}$$

which is minimized instead of the misclassification error in (1.6). The formula may be adapted for variables with m categories to replace the misclassification error in (1.5):

$$MC_m(c_{i,j}) := \frac{\sum_{i,j:i\neq j} c_{i,j}F_{i,j}}{N} \tag{1.19}$$

but it requires the evaluation of $m(m-1)$ types of misclassification. Finally, if the target variable is numerical, we can punctually weight the residuals. For instance, the mean absolute error in (1.14) will be adjusted as follows:

$$\text{mean absolute error}(w_1, \ldots, w_N) := \frac{1}{N}\sum_{i=1}^{N} w_i|u_i|. \tag{1.20}$$

Before concluding this section, it is worth highlighting the scope of all the aforementioned performance measures. In traditional statistics, performance measures such as the sum of squared errors are merely used as a

target function to minimize, in order to guarantee the best possible fit in regression models. In machine learning, by contrast, a performance measure is not only used as a target function in the training step, but also as a validation tool to pick the best model. In analytics, there is no *a priori* superior model, so we should always train a number of algorithms by minimizing some error function, and then choose the algorithm returning the smallest *validation* error computed with the same error measure. We will proceed this way throughout the dissertation.

1.4.3 Model selection criteria

In Subsection 1.4.2, we focused on the main reason why machine learning tools are so useful: *accuracy*. Generally speaking, the accuracy of a model is represented by its error on out-of-sample data, that is, the validation error. As we will see in our applications, machine learning algorithms tend to outperform traditional parametric techniques. However, accuracy should not be alone when it comes with comparing predictive models to select the best one. Hastie et al. (2009) lists a number of general criteria that should be taken into account when selecting a model over others. We will focus on those criteria, although there may be reasons to ignore some of them or enhance the list itself.

Accuracy being equal, *stability* is what makes the difference, especially in machine learning. Stability may be quantitatively defined by the gap between the training error and the validation error, which is usually positive: the larger the gap, the lower the stability. In general, the more an algorithm is accurate, the less it is stable, in the sense that this accuracy may disappear in predicting out-of-sample data. Once the algorithm is trained to return the best accuracy, then it is tuned on the validation dataset in order to avoid instability (i.e., overfitting), but maintaining a reasonable level of accuracy as well. In any case, we will never be able to maximize accuracy and stability at the same time, so a data scientist will always deal with a *accuracy-stability trade-off*.

Beyond the measurable features of a predictive model, we have some formal features as well. In practice, they are very important, although there is no measure for them. First of all, *flexibility* is a quite desirable characteristic. Generally speaking, a model is flexible if it is complex enough to capture soft information and weak dependencies with minimal risk of instability and overfitting. For instance, this is the case of polynomial regressions and neural networks, which allow for nonlinear relationships among variables. This is not surprising, in light of the high number of parameters they depend on. On the other hand, even if a model is flexible, there is however a way to tune it in order to avoid instability. For instance, we can reduce the terms of a polynomial regression as well as the hidden neurons of a neural network. Nonetheless, flexibility is not only about the number of parameters within a

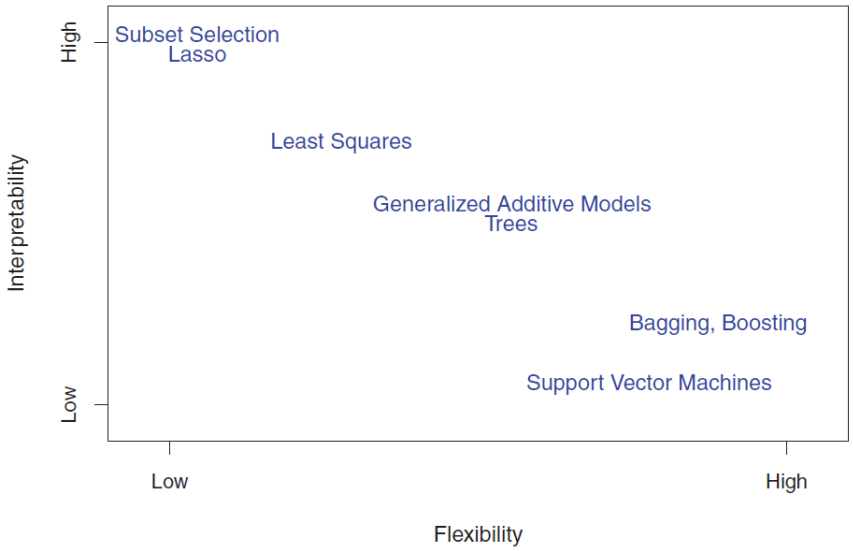


Fig. 1.10: Flexibility-interpretability trade-off (see Hastie et al. (2009))

model, but it is also about applicability to both classification and regression, or adaptability to both categorical and numerical variables - just to mention a couple of examples.
Together with flexibility, an ideal model should be characterized by *interpretability* too. A model is interpretable if one can easily explain its outcome with minimal risk of misunderstanding it. In this case, regressions represent the main example of highly interpretable models. Since they are based on mere multiplicative parameters into some regular function (e.g., sums, logs, etc.), result interpretation and communication will never be a problem. Unfortunately, high interpretability often means low flexibility and vice versa. A linear regression is extremely simple to understand, but it is rarely flexible enough to explain variance in a fairly complex dataset. On the other hand, when flexibility is high, with structured functions and many parameters, it is difficult to distinguish the various impacts. In some cases (e.g., neural networks), you face the typical "black-box" effect, that is, you feel like the model returns very accurate predictions, but you don't know why and lose control on them. To summarize, we face a further trade-off, the *flexibility-interpretability trade-off*. In Figure 1.10, it is reported a representation of such a trade-off, using some machine learning methods. In particular, we observe that regression-based models (i.e., least squares, generalized additive models, Lasso, etc.) lay on the left - low flexibility and high interpretability - whereas common machine learning tools such as bagging, boosting, and support vector machines lay on the right - high flexibility and low interpretability.

To conclude the section, it is worth mentioning a last, broad feature of a good predictive model: *implementability*. This is a rather soft concept that may be defined as the effective applicability of the model in the light of necessary data versus available data, computational tractability of the underlying algorithm, IT infrastructure at disposal, and any other practical requirements. As an example, clustering methods are some of the most common models in machine learning, widely used in academia and industry. They may require the computation of a distance function - whatever it is - among all the records in a dataset. If a dataset contains 100 thousands records (a fairly large dataset, but no big data), we will nearly compute and save 5 billion distances. In many commercial computers, this is impossible. Clustering methods are quite attractive, but they may result in implementability issues. More importantly, there may be problems in implementing a model in a real business situation. Insurance industry is highly regulated, and machine learning tools may be problematic in the real world. This is a major issue, perhaps the main issue for actuaries in business intelligence.

1.5 Statistics versus machine learning

So far, we gave a fair amount of information about machine learning and analytics. Some concepts are similar - or even equivalent - to what traditional statistics encompass. For instance, the distinction between classification, prediction and forecasting (see Section 1.1) can be done in statistical methodology as well, e.g., logistic regressions classify, linear regressions predict, and autoregressive models forecast. Moreover, some of the performance measures we introduced in Subsection 1.4.2 are broadly used in statistics as well, for instance, the sum of squared errors in regression and the ROC curve in classification.
Apparently, statistics and machine learning are quite similar, but, as stated in Charpentier (2015a),

> *The goal for a statistician is to predict an interaction between variables with some degree of certainty (we are never 100% certain about anything). Machine learners, on the other hand, want to build algorithms that predict, classify, and cluster with the most accuracy.*

Actually, the goal changes everything, and makes machine learning deeply different to statistics.
First, the theory. In traditional statistics, probabilistic assumptions lock the model in a predefined framework in order to ensure useful asymptotic properties and other theoretical advantages. For example, we know that the estimator of the ordinary linear regression is the best possible estimator *under some assumptions*, as we will recall later in this chapter. By contrast,

any machine learning technique is based on an algorithm rather than an underlying model. As long as the algorithm converges to an accurate estimation of the target variable, we are not really interested in a theoretical justification of the code behind the algorithm.
Second, the data. In traditional statistics, data gets less importance than in machine learning because of the theoretical assumptions. Since statistical models often assume some analytical form for the distribution of the target variable, any deviation from that distribution is not even considered in practice. If such a deviation is actually recognized in data, it will be probably addressed by another separate model. A quite common example is in actuarial pricing, where the outliers are excluded from a dataset in order to let the regression work; those outliers are then treated separately. On the other hand, machine learning algorithms may be tailored to meet data features at best. The advantage is that they can potentially learn from any type of dataset and predict any type of target variable, but the disadvantage is that they are heavily dependent on data. This drawback is partially solved by the usage of out-of-sample data - the validation data and/or the test data - but it generally comes from the original population of data. In other words, the validation of a machine learning algorithm is essentially a validation conditioned to that data, but there is no reason to believe that the same customized algorithm will work, for instance, with a analogue dataset coming from a different source.
Third, the parameters. In traditional statistics, parameters mostly describe distributions. For instance, the estimated parameters in linear regression can be seen as features of the distribution of the real parameters. In machine learning, it is not that easy. First and foremost, there are few pure nonparametric methods which do not need any parameter. Most of the algorithms, however, will include some structural parameters, or *meta-parameters*, that is, parameters used to calibrate (rather than train) the algorithm. For instance, the nearest neighbours algorithm is defined by one meta-parameter, that is, the number of data points that lay within the neighbourhood of a record. Other algorithms are considered semiparametric, in the sense that they include both meta-parameters and real functional parameters. Neural networks can be considered semiparametric since their architecture is based on weights (i.e., the parameters) estimated on the training dataset, while the number of the hidden neurons or the learning rate (i.e., the meta-parameters) should be properly predefined. To summarize, parameters are actually trained to optimize the predictive performance, whereas meta-parameters are chosen depending on heuristic criteria.
Fourth, the estimation error. Consider the generic model

$$Y = \phi(X) + \nu \tag{1.21}$$

where the explanatory variable X describes the target variable Y through some deterministic function ϕ and zero-mean noise ν. If f estimates ϕ by an

optimization algorithm that minimizes the quantity $E[(Y - f(X))^2]$, then

$$\widehat{Y} = f(X) \tag{1.22}$$

where f is a random variable, and the estimation error is

$$\begin{aligned} u^2 &:= E[(Y - \widehat{Y})^2] = E[(Y - f(X))^2] = \\ &= E[Y^2 + f(X)^2 - 2Yf(X)] = \\ &= E[Y^2] + E[f(X)^2] - 2E[Yf(X)] = \\ &= Var(Y) + Var(f(X)) + E[Y]^2 + E[f(X)]^2 - 2E[Yf(X)] = \\ &= Var(\nu) + Var(f(X)) + \phi(X)^2 + E[f(X)]^2 - 2\phi(X)E[f(X)] = \\ &= \underbrace{Var(\nu)}_{\text{noise}} + \underbrace{Var(f(X))}_{\text{variance}} + \underbrace{(\phi(X) - E[f(X)])^2}_{\text{bias}} \end{aligned} \tag{1.23}$$

which is the so-called *bias-variance decomposition*. Bias refers to the systematic error due to the approximation of a real-world, complicated functional form ϕ with a simpler functional form f. For instance, estimating $\phi(X) = a + bX + cX^2$ through $f(X) = \alpha + \beta X$ will always introduce bias because of the fundamental difference in shapes. Variance refers to the error due to the randomness of the sample used to train the model and return f. Since we will never use the entire population as training dataset, f is a function of the training data, and it is a random variable with its own variance. As a general rule, we can say that

- the higher the bias, the higher the potential of *underfitting* (i.e., missing signal)
- the higher the variance, the higher the potential of *overfitting* (i.e., capturing noise).

An ideal model should minimize both, in order to explain regularities in the training dataset and generalize well in the validation dataset. However, this is generally impossible since underfitting and overfitting are opposite concept. This leads to a formal separation between high-bias low-variance models and low-bias high-variance models. In traditional statistics, parametric methods such as linear regression or logistic regression with a limited number of parameters seldom overfit: they are quite strong in terms of variance reduction, although their static functional assumptions increase the bias. This is the reason why out-of-sample analysis with validation and test data in less relevant in regression. In machine learning, on the other hand, algorithms tend to be quite flexible, and flexibility generally implies better fitting - low bias - but poor generalization - high variance. For instance, in neural networks, the more the hidden neurons, the more the weights and parameters used in the architecture: that increases flexibility, but introduces a major risk of overfitting. As a consequence, a machine learning algorithm is

not only built to return more accurate estimations with lower bias, but also to limit the increase in variance by testing the model on out-of-sample data. All in all, we should choose a model able to reduce bias and limit variance at the same time - the so-called *bias-variance trade-off*.

1.6 Machine learning tools

This section is supposed to provide the reader with a fair overview of the data scientist toolkit. Starting from the classical generalized linear models in Subsection 1.6.1, we will quickly go through the main machine learning techniques (see Subsection 1.6.2) in data science, and the most widespread software packages used to implement them (see Subsection 1.6.3). Even if some of those topics will be further discussed later in the dissertation, it is important to realize the great variety that characterizes data science.

1.6.1 A first example: GLMs

Regression techniques such as generalized linear models (GLMs) are by far the most used fitting tools in industry. They rely on a unique combination of theoretical solidity and practical interpretability, which makes them one of the favourite tools among actuaries. For the same reasons, regression models do not represent a powerful machine learning tool. Unfortunately, the assumption set they require causes some rigidity that limits the ability of learning from data. Nonetheless, since we are going to use two common GLMs such as logistic regression and gamma regression later on, it is worth providing a brief description, revisiting and drawing on those of De Jong et al. (2008).
Very generally, we can assume that the target variable y is a function of some predictors, that is, the explanatory variables of the i^{th} record:

$$y_i := \phi(x_{i1}, \dots, x_{in}), \qquad \forall i. \tag{1.24}$$

If we can rely on an algorithm that estimates ϕ by building a new regular function f, we will get an estimation of the target variable:

$$y_i \simeq f(x_{i1}, \dots, x_{in}) + u_i \quad \Longrightarrow \quad \widehat{y}_i = \widehat{f}(x_{i1}, \dots, x_{in}) \tag{1.25}$$

where u_i denotes the estimation error (notice the similarities between (3.3) and (1.25)). Quite intuitively, the function f will be somehow estimated in order to minimize the errors u_i, or, more precisely, a proper combination of them representing the overall error of the model. However, the shape of f should be defined as an assumption. In *multiple linear regression*, for instance, its shape is linear:

$$y_i = \beta_0 + \sum_{j=1}^{n} x_{ij}\beta_j + u_i \quad \Longrightarrow \quad \widehat{y}_i = \widehat{\beta}_0 + \sum_{j=1}^{n} x_{ij}\widehat{\beta}_j \tag{1.26}$$

where β_j denotes the estimation parameter related to the j^{th} predictor. Remember that the linearity refers to the coefficients $\beta_0, \ldots, \beta_n$ rather than to the predictors $x_{i1}, \ldots, x_{in}$. In other terms, we can use the predictors as they appear in the dataset as well as any transformation or interaction of them, linear or nonlinear: in any case, the model will still be linear. Defining

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}, \boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & \ldots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \ldots & x_{Nn} \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_n \end{pmatrix}, \boldsymbol{u} = \begin{pmatrix} u_1 \\ \vdots \\ u_N \end{pmatrix} \tag{1.27}$$

the model (1.26) may be also expressed in a matricial form:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{u} \quad \Longrightarrow \quad \widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}. \tag{1.28}$$

Beside the shape of f, the followings should hold too:

1. $rank(\boldsymbol{X}) = n$
2. $E[\boldsymbol{u}] = \boldsymbol{0}$
3. $Var[\boldsymbol{u}] = \sigma^2 \boldsymbol{I}_N$

where $\boldsymbol{I}_N$ denotes the identity matrix of size N. In other words, the rank of $\boldsymbol{X}$ is the maximum possible rank so that no redundant information is there (assumption 1.), the random variables of the estimation error are null on average (assumption 2.), and they are also independent and homoscedastic with variance σ^2 (assumption 3.). In particular, the assumption 2. directly implies the followings:

2a. $E[\boldsymbol{y}] = E[\boldsymbol{y}|\boldsymbol{X}] = \boldsymbol{X}\boldsymbol{\beta} = \widehat{\boldsymbol{y}}$

2b. $Var(\boldsymbol{y}) = Var(\boldsymbol{y}|\boldsymbol{X}) = \sigma^2 \boldsymbol{I}_N$.

Once the assumptions have been verified, the vector of parameters $\boldsymbol{\beta}$ can be estimated through the *least squares method*. It is based on the minimization of the sum of squared error u_i^2:

$$\begin{aligned} Q(\boldsymbol{\beta}) &:= \sum_{i=1}^{N} u_i^2 = \sum_{i=1}^{N} (y_i - \widehat{y}_i)^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) = \\ &= \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} = \\ &= \boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \end{aligned} \tag{1.29}$$

which is a quadratic function, thus its minimum is necessarily its only stationary point, that is, the solution of the following system:

$$\frac{\partial Q(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \frac{\partial}{\partial \boldsymbol{\beta}} \Big(\boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \Big) = -2\boldsymbol{X}^T\boldsymbol{y} + 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} = 0 \tag{1.30}$$

that is

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{1.31}$$

Using the second equation of (1.28), we also get

$$\widehat{\boldsymbol{y}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{1.32}$$

and using the first equation of (1.28)

$$\widehat{\boldsymbol{u}} = \boldsymbol{y} - \widehat{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \Big[\boldsymbol{I}_N - \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\Big]\boldsymbol{y}. \tag{1.33}$$

Actually, we got a model now: we are able to calculate $\widehat{\boldsymbol{y}}$ by using $\widehat{\boldsymbol{\beta}}$ in the second equation of (1.28), as long as all the aforementioned assumptions hold. However, the traditional multiple linear regression includes a further, crucial assumption:

$$\boldsymbol{u} \sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I}_N) \tag{1.34}$$

which implies

$$\boldsymbol{y} \sim N(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{I}_N). \tag{1.35}$$

Thanks to the normality of the vector $\boldsymbol{u}$ of the residuals, a whole range of relevant statistics can be deduced from the model, including parameter distributions, confidence interval and so on. Among others, for instance, it can be easily proven that $\widehat{\boldsymbol{\beta}}$ is normally distributed with mean $\boldsymbol{\beta}$, and $\widehat{\boldsymbol{y}}$ is normally distributed with mean $\boldsymbol{y}$.

Once the model has been defined, we want an effective way to evaluate its performance. The theoretical framework helps us to define a rather intuitive but rigorous tool for performance evaluation. First, the *total deviance* is defined as

$$D_T(\boldsymbol{y}) := \sum_{i=1}^{N}(y_i - E[\boldsymbol{y}])^2 \tag{1.36}$$

and it represents a measure of the information contained into the data. It may be further manipulate as follows:

$$\begin{aligned} D_T(\boldsymbol{y}) &= \sum_{i=1}^{N}(y_i - \widehat{y}_i + \widehat{y}_i - E[\boldsymbol{y}])^2 = \sum_{i=1}^{N}(y_i - \widehat{y}_i)^2 + \sum_{i=1}^{N}(\widehat{y}_i - E[\boldsymbol{y}])^2 =: \\ &=: D_E(\boldsymbol{y}) + D_R(\boldsymbol{y}) \end{aligned} \tag{1.37}$$

where the *explained deviance* $D_E(\boldsymbol{y})$ measures the information explained by the model, while the *residual deviance* $D_R(\boldsymbol{y})$ measures the residual information that the model could not detect. Naturally, the greater $D_E(\boldsymbol{y})$, the smaller $D_R(\boldsymbol{y})$, the better the model. In fact, the *coefficient of determination* - known as *R-squared* and defined by the deviance measures - is the most common performance evaluation tool for multiple linear regression:

$$R^2(\boldsymbol{y}) := \frac{D_E(\boldsymbol{y})}{D_T(\boldsymbol{y})} = 1 - \frac{D_R(\boldsymbol{y})}{D_T(\boldsymbol{y})} \tag{1.38}$$

which is a value in $[0, 1]$. Once again, the greater $D_E(\boldsymbol{y})$, the greater $R^2(\boldsymbol{y})$, the better the model, but the R^2 has the additional advantage of the normalization. In practice, it means that the R^2 makes possible a fair performance comparison among different models. However, it does not take into account the simplicity of the model as measured by the number of explanatory variables. In other terms, we should still prefer a model with more variables as long as the increase in R^2 is sufficient to explain the greater complexity. Therefore, statisticians often adjust the R^2 as follows:

$$R^2_{adj}(\boldsymbol{y}) := 1 - \frac{(1 - R^2(\boldsymbol{y}))(N - 1)}{N - n - 1} \tag{1.39}$$

to penalize the R^2 by the number of predictors.
Using (1.35), the model definition (1.28) may be written differently:

$$E[\boldsymbol{y}] = \boldsymbol{X}\boldsymbol{\beta} \quad \Longrightarrow \quad E[\widehat{\boldsymbol{y}}] = \boldsymbol{X}\widehat{\boldsymbol{\beta}}. \tag{1.40}$$

which is a direct consequence of the normality assumed. However, if we relax that assumption, it can be more generally assumed that there is a regular (i.e., invertible and derivable) *link function* g - different to the identity function - mapping $E[\boldsymbol{y}]$ to $\boldsymbol{X}\boldsymbol{\beta}$:

$$g(E[\boldsymbol{y}]) = \boldsymbol{X}\boldsymbol{\beta} \tag{1.41}$$

or equivalently

$$E[\boldsymbol{y}] = g^{-1}(\boldsymbol{X}\boldsymbol{\beta}). \tag{1.42}$$

The main reason why such a generalization is very useful in practice is that g maps from some subset $X \subseteq \mathbb{R}$ to $\mathbb{R}$, that is, g^{-1} transforms the linear predictor $\boldsymbol{X}\boldsymbol{\beta} \in \mathbb{R}$ to the target variable prediction in X. In fact, generalized regression is able to handle target variables defined in a specific subset of $\mathbb{R}$. This is not possible in multiple linear regression, unless we operate a proper transformation of the target variable itself (this is sometimes enough, but it introduces *transformation bias* into the model). For instance, if the target variable represents a positive amount, the quickest ways to use regression are

- convert amounts to logarithmic amounts, predict the latter through multiple linear regression, and convert the predictions back by using the exponential function;
- choose a link function $g : (0, +\infty) \to \mathbb{R}$, and predict the amounts through the related generalized regression.

Actually, the choice of the link function is not direct, rather it is a consequence of the distribution we assume for the target variable. This is possible

since generalized regression assumes that such a distribution belongs to the exponential family, whose density is:

$$f_e(y_i;\theta_i,\phi) := e^{\frac{y_i\theta_i - c(\theta_i)}{\phi} + h(y_i,\phi)} \tag{1.43}$$

where θ_i denotes the *canonical parameter* varying by observation, while ϕ denotes the *dispersion parameter* that is constant for all observations. In other words, the distribution constraint is not completely eliminated, rather it is "generalized" to a wider range of distributions - a distribution family. When it comes with prediction of amounts, whose distributions are often nonnegative and heavy-tailed, a common choice is the gamma distribution (for instance, see Charpentier (2015b)), defined by a *shape parameter* $\varphi > 0$ and a *scale parameter* $\vartheta_i > 0$:

$$f_\Gamma(y_i;\vartheta_i,\varphi) := \frac{y_i^{\varphi-1} e^{-\frac{y_i}{\vartheta_i}}}{\vartheta_i^{\varphi}\Gamma(\varphi)}. \tag{1.44}$$

which can be easily rearranged as follows:

$$f_\Gamma(y_i;\vartheta_i,\varphi) = e^{-\frac{y_i}{\vartheta_i} - \varphi\ln\vartheta_i + (\varphi-1)\ln y_i - \ln\Gamma(\varphi)}. \tag{1.45}$$

If we define $\theta_i := -\frac{1}{\varphi\vartheta_i}$ and $\phi := \frac{1}{\varphi}$, then

$$\begin{aligned} f_\Gamma(y_i;\vartheta_i,\varphi) &= e^{\frac{y_i\theta_i}{\phi} - \frac{1}{\phi}\ln\left(-\frac{1}{\varphi\theta_i}\right) + \left(\frac{1}{\phi}-1\right)\ln y_i - \ln\Gamma\left(\frac{1}{\phi}\right)} = \\ &= e^{\frac{y_i\theta_i}{\phi} - \frac{1}{\phi}\ln\left(-\frac{1}{\theta_i}\right) + \left(\frac{1}{\phi}-1\right)\ln y_i + \varphi\ln\varphi - \ln\Gamma\left(\frac{1}{\phi}\right)} = \\ &= e^{\frac{y_i\theta_i - \ln\left(-\frac{1}{\theta_i}\right)}{\phi} + \left(\frac{1}{\phi}-1\right)\ln y_i - \frac{1}{\phi}\ln\phi - \ln\Gamma\left(\frac{1}{\phi}\right)} \end{aligned} \tag{1.46}$$

which belongs to the exponential family in (1.43) with

$$c(\theta_i) := \ln\left(-\frac{1}{\theta_i}\right), \qquad h(y_i,\phi) := \left(\frac{1}{\phi}-1\right)\ln y_i - \frac{1}{\phi}\ln\phi - \ln\Gamma\left(\frac{1}{\phi}\right). \tag{1.47}$$

This is important because it implies that

$$E[y_i] = \frac{dc(\theta_i)}{d\theta_i} = \frac{d}{d\theta_i}\ln\left(-\frac{1}{\theta_i}\right) = -\frac{d}{d\theta_i}\ln\theta_i = -\frac{1}{\theta_i} = \varphi\vartheta_i \tag{1.48}$$

$$Var(y_i) = \phi\frac{d^2c(\theta_i)}{d\theta_i^2} = -\phi\frac{d}{d\theta_i}\frac{1}{\theta_i} = \frac{\phi}{\theta_i^2} = \varphi\vartheta_i^2. \tag{1.49}$$

All in all, how should we define the link function starting from these considerations? First, notice that the normal distribution belongs to the exponential family too, because

$$f_N(y_i;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-\mu)^2}{2\sigma^2}} = \ldots = e^{\frac{y_i\mu - \frac{\mu^2}{2}}{\sigma^2} - \frac{y_i^2}{2\sigma^2} - \ln\sqrt{2\pi\sigma^2}} \tag{1.50}$$

thus we should define $\theta_i := \mu$ and $c(\theta_i) := c(\mu) = \frac{\mu^2}{2}$, so that

$$E[y_i] = \frac{dc(\mu)}{d\mu} = \mu = \theta_i. \tag{1.51}$$

Actually, if the target variable is normally distributed, we will be back to the multiple linear regression. In such a case, we may use (1.40) to write:

$$E[y_i] = \theta_i \tag{1.52}$$

so a generalized regression model might be simply redefined as

$$E[y_i] = g^{-1}(\theta_i). \tag{1.53}$$

At the same time, if the target variable is gamma distributed, we also know that $E[y_i] = \varphi\vartheta_i$ and $\theta_i = -\frac{1}{\varphi\vartheta_i}$, so

$$g(\varphi\vartheta_i) = -\frac{1}{\varphi\vartheta_i} \tag{1.54}$$

which implies

$$E[y_i] = \varphi\vartheta_i = g^{-1}(\boldsymbol{x}_i^T\boldsymbol{\beta}) = -\boldsymbol{x}_i^T\boldsymbol{\beta} = -\beta_0 - \sum_{j=1}^{n}\beta_j x_{ij}. \tag{1.55}$$

This is a somewhat natural choice, coming directly from the theory, thus it is called *canonical link function*. For GLMs, it is always possible to define g in a canonical way, that is, imposing (1.52), and this choice implies a range of desirable features in our model. However, others link functions might be rather used, and sometimes there are good reasons to.
Even if the gamma distribution is a common choice in the actuarial practice, we should highlight that this is not the only one. First of all, we may choose other asymmetric and nonnegative distributions from the exponential family (e.g., inverse Gaussian) for the GLM. Moreover, we may use the so-called *Tweedie distributions* (see Jørgensen (1987)), which represent a subfamily of the exponential family, assuming

$$Var(y_i) = \phi E[y_i]^p. \tag{1.56}$$

where p is nonnegative. For $p = 2$, as an example, the first equation in (1.48) implies

$$Var(y_i) = \phi\left(-\frac{1}{\theta_i}\right)^2 = \frac{\phi}{\theta_i^2} \tag{1.57}$$

which matches the second equation in (1.48). In other words, the Tweedie distribution with $p = 2$ is equivalent to the gamma distribution. Likewise, we could easily prove that $p = 0$, $p = 1$ and $p = 3$ return the normal

distribution, the Poisson distribution and the inverse Gaussian distribution. For any other $p > 1$, we obtain new Tweedie distributions for our GLM. Given that it includes a further degree of freedom, we may expect higher accuracy.

Probabilities concern actuaries as much as amounts: they need a GLM to cope with them too. *Logistic regression* is not really an "advanced" tool, but it is worthwhile to describe it here since it represents the most used tool to fit and predict probabilities such as surrender rates (for example, see Cerchiara et al. (2009), Kiesenbauer (2012) and Kim (2010)). Obviously, we should choose a link function $g : (0, 1) \to \mathbb{R}$. This is quite straightforward if assume that the target variable is distributed as a Bernoulli with parameter p_i. First, notice that the Bernoulli distribution is

$$\begin{aligned} f_B(y_i; p_i) &:= p_i^{y_i}(1-p_i)^{1-y_i} = e^{y_i \ln p_i + (1-y_i)\ln(1-p_i)} = \\ &= e^{y_i \ln\left(\frac{p_i}{1-p_i}\right) + \ln(1-p_i)} = e^{y_i \ln\left(\frac{p_i}{1-p_i}\right) - \left[\ln\left(\frac{p_i}{1-p_i}\right) - \ln p_i\right]} \end{aligned} \tag{1.58}$$

and if we define $\theta_i := \ln\left(\frac{p_i}{1-p_i}\right)$ and $\phi := 1$, then

$$f_B(y_i; \theta_i) = e^{y_i\theta_i - \left[\theta_i - \ln\left(\frac{1}{1+e^{-\theta_i}}\right)\right]} \tag{1.59}$$

which belongs to the exponential family in (1.43) with

$$c(\theta_i) := \theta_i - \ln\left(\frac{1}{1+e^{-\theta_i}}\right), \qquad h(y_i, \phi) := 0. \tag{1.60}$$

In particular, it implies that

$$\begin{aligned} E[y_i] &= \frac{dc(\theta_i)}{d\theta_i} = \frac{d}{d\theta_i}\left[\theta_i - \ln\left(\frac{1}{1+e^{-\theta_i}}\right)\right] = \frac{1}{1+e^{-\theta_i}} = \\ &= p_i \end{aligned} \tag{1.61}$$

$$\begin{aligned} Var(y_i) &= \phi\frac{d^2c(\theta_i)}{d\theta_i^2} = \frac{d}{d\theta_i}\frac{1}{1+e^{-\theta_i}} = \frac{e^{-\theta_i}}{(1+e^{-\theta_i})^2} = \\ &= p_i(1-p_i) \end{aligned} \tag{1.62}$$

which is exactly what we expect from a Bernoulli distribution.

Using (1.52), the canonical link function of the logistic regression is such that

$$g(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) \tag{1.63}$$

which defines the so-called *logit function*. In other words, the probability equals

$$E[y_i] = p_i = g^{-1}(\boldsymbol{x}_i^T\boldsymbol{\beta}) = \frac{1}{1+e^{-\boldsymbol{x}_i^T\boldsymbol{\beta}}} = \frac{1}{1+e^{-\beta_0 - \sum_{j=1}^n \beta_j x_{ij}}} \tag{1.64}$$

defining the *logistic function*. Even if it represents the canonical link as well as the most widely used choice, other link functions are still available (e.g., the *probit*, often used in econometrics, and the *CLogLog*, more common in survival analysis).
Regardless of the underlying distribution of a GLM, we cannot use the ordinary least square method to estimate the vector $\widehat{\boldsymbol{\beta}}$. Rather, we should use the maximum likelihood estimation. For instance, the probability estimated by the logistic regression equals:

$$\widehat{p_i} := g^{-1}(\boldsymbol{x}_i^T \widehat{\boldsymbol{\beta}}) = \frac{1}{1 + e^{-\boldsymbol{x}_i^T \widehat{\boldsymbol{\beta}}}} = \frac{1}{1 + e^{-\widehat{\beta}_0 - \sum_{j=1}^{n} \widehat{\beta}_j x_{ij}}}. \tag{1.65}$$

The quality of GLMs such as gamma regression and logistic regression can be still assessed by using proper modified versions of the R^2_{adj} in (1.39). Nonetheless, there are other common quality measures such as the *Akaike Information Criterion* (see Akaike (1973)) and the *Bayesian Information Criterion* (see Schwarz (1978)):

$$AIC := 2n - 2\ln \hat{L} \tag{1.66}$$

$$BIC := n\ln N - 2\ln \hat{L} \tag{1.67}$$

where $\hat{L}$ is the maximum value of the likelihood function for the model. The main problem of AIC and BIC is that they do not give any indication about the absolute quality of the regression, while R^2 does. AIC and BIC may be used to compare regression models from the same dataset, accounting for penalizations due to complexity ($2n$ for AIC and $n \ln N$ for BIC), but they do not reveal anything about the goodness of fit of one single model. All in all, AIC and BIC may be used for model selection only.
Indeed, at the beginning of this subsection, we implicitly assumed to know the range of explanatory variables $x_1, \ldots, x_n$. Of course, we know the explanatory variables in the dataset, but how should we select them as $x_1, \ldots, x_n$? Because of multicollinearity among potential explanatory variables, we cannot simply run the regression on all of them, and then select only the most significant ones based on their p-values. Rather, we should somehow select different sets of explanatory variables and run the related regression: the model with the highest R^2_{adj} or the lowest AIC or BIC will be selected. The different sets of explanatory variables depend on the algorithm used to select them. There are mainly three popular iterative search algorithms.
In *forward selection*, we start with no predictors, and then add them one by one. Each added predictor is that (among all predictors) with the largest contribution to the goodness of fit on top of the predictors that are already in it. The algorithm stops when the contribution of additional predictors is not statistically significant.
In *backward selection*, we start with all predictors, and then eliminate the

least useful one at each step according to statistical significance. The algorithm stops when all the remaining predictors have significant contributions. Finally, *stepwise selection* is like forward selection except that at each step we consider dropping predictors that are no longer statistically significant, as in backward selection. In all our applications where regressions are involved, our data will be always regressed through stepwise selection.
To finally conclude the section, some practical remarks are to be highlighted. Whatever the specific regression model, its assumptions make it theoretically strong, but VERY weak from a practical perspective. Actually, We may condense them with the following, practical premises:

- no allowance for unanticipated relationships (i.e., high bias)
- no allowance for dependencies among predictors
- no allowance for outliers.

They somewhat simplify the original assumptions, but give a fair idea of what those assumptions really mean, that is, we can hardly expect good performance from regression methods when information is far to be regular. Data is generally affected by missing values, correlations, heavy tails, asymmetries, nonlinearities and any other type of distortion. These are the reasons why regressions do not always represent the best compromise, whereas data-driven techniques may offer a further option.

1.6.2 Beyond regression

One of the goals of this dissertation is to provide actuaries with a practical introduction of the most common machine learning methods, stepping out the traditional regression framework. We will look more deeply into them later, but this subsection will give an all-in overview, beyond the mere distinction between unsupervised and supervised learning.
A first, simple unsupervised learning method is the so-called *association rules*, based on an algorithm, known as *Apriori algorithm*, taking advantage of *a priori* likelihoods. Essentially, it uses the concept of conditional probability to detect interesting and significant relations between variables in large databases. Association rules will be discussed in Subsection 2.3.1.
More commonly, unsupervised learning is part of the data preparation process before any prediction with supervised learning. In particular, it is used for dimension reduction, which is a major issue when it comes with huge amounts of data. It is not only about generic infrastructural problems in handling it, but also computational limits of algorithms and/or an unfavourable number of records in relation to the number of variables.
For instance, if we need to reduce the fields in a dataset, then *principal component analysis* (PCA) for numerical variables and *multiple correspondence*

analysis for categorical variables represent very common choices in data science. Both of them take in input the original fields (more specifically, PCA uses the covariance matrix, while multiple correspondence analysis uses the contingency table) in order to generate new, uncorrelated variables revealing underlying relationships and supporting feature selection. In particular, PCA will be discussed in Subsection 2.3.2.
Rather than "columns", one could be interested in reduce "rows", that is, the records themselves. Broadly speaking, *cluster analysis* is what we need. It encompasses a huge number of different methods and adaptations, so we will only mention few of them. Although cluster analysis is a widely accepted approach in dimension reduction, its major drawback relates to the fact that there is no "objectively" correct clustering algorithm. Traditional algorithms are often adjusted to fit specific problems, while there is seldom a mathematical reason to prefer one method over others. In addition, given that there is no prediction accuracy as a performance reference (in fact, there is no target variable to predict), clustering methods cannot be compared with each other on such a basis. Most of the time, we should already have an idea about the clusters we expect, and the best methods will be those detecting them. To quote Estevill-Castro (2002)

> *Clustering is in the eye of the beholder.*

Two of the most common clustering methods are *hierarchical clustering* and *K-means clustering*, described in Subsection 2.3.3. Both of them are *distance-based algorithms*, that is, they group records by using a predetermined distance function that measures similarity. In spite of their simplicity, these algorithms force the points into areas defined by convex boundaries which do not represent the natural growth of clusters. Another drawback is about the choice of the number of clusters, which is always up to the data scientist and hardly suggested by the algorithms and their outcomes.
As a consequence, *distribution-based algorithms* are sometimes preferred for their versatility, in spite of the complexity they introduce. For instance, they encompass the *Gaussian mixture models*, which model the data with a fixed number of Gaussian distributions, whose parameters are iteratively optimized to better fit the dataset. Another, more versatile category of cluster methods encompasses *density-based algorithms* such as DBSCAN (see Ester et al. (1996)) and OPTICS (see Ankerst et al. (1999)), which group data points accounting for a given density tolerance, so that only high-density group of points are actually considered as clusters. In this way, only significant clusters are returned, and outliers are automatically excluded.
The supervised learning world is even more diverse than that of unsupervised learning. Traditionally, GLMs represent the first choice to actuaries. If applied together with a variable selection algorithm such as stepwise, their main strength is the robustness, that is, low variance and little overfitting. However, this is not always the case. For instance, in a dataset where records

are few as compared to the number of features, even GLMs are likely to overfit. To solve the problem, we may use *regularized linear models* (RLMs) in order to avoid the excessive growth of the regression parameters. To do that, we may properly adjust the objective function of GLMs. More specifically, RLMs minimize the objective function defined by the usual sum of squared errors plus the correction $R(\boldsymbol{\lambda}, \boldsymbol{\beta})$, that is, a predetermined *penalty function* of the learning rate vector $\boldsymbol{\lambda}$ and the parameter vector $\boldsymbol{\beta}$. Among the most common choices for $R(\cdot, \cdot)$, it is worth mentioning the followings:

- *Ridge regression*: $R(\boldsymbol{\lambda}, \boldsymbol{\beta}) := \lambda \sum_i \beta_i^2$
- *Lasso regression*: $R(\boldsymbol{\lambda}, \boldsymbol{\beta}) := \lambda \sum_i |\beta_i|$
- *Elastic net*: $R(\boldsymbol{\lambda}, \boldsymbol{\beta}) := \lambda_1 \sum_i |\beta_i| + \lambda_2 \sum_i \beta_i^2$.

As we said, RLMs can be useful in rather rare situations that make GLMs unstable, but that is not the case in the actuarial applications of the next chapters, so we will not look into them further.
Another common parametric method is the *linear discriminant analysis* (LDA), which can be seen as a modification of PCA to solve classification problems. LDA works by creating one or more linear combinations of predictors, creating a new latent variable for each of them. The first variable created, say v_1, maximizes the differences between groups on v_1. The second variable, say v_2, maximizes differences on v_2, but also must not be correlated with v_1. The third variable, say v_3, maximizes differences on v_3, but also must not be correlated with v_1 and v_2. This continues with the requirement that the new variable should not be correlated with any of the previous variables.
Just like RLMs represent a machine-learning-oriented modification to GLMs, *support vector machines* (SVMs) represent a machine-learning-oriented modification to LDA. Indeed, LDA assumes a number of hypotheses - the same as those of linear regression - in order to admit a single, analytical solution. This makes LDA strong in theory, but rigid in practice. SVM enhances LDA in two ways. First, it makes use of a "slack variable" that allows a certain amount of overlap between the groups, in order to avoid the rigid inclusion of any data point - even potential outliers - in one of them. Second, it makes use of kernel functions in order to allow for non-linearities by transforming the basic linear classifier to a non-linear classifier.
Moving away from parametric models such as GLM, RLM, LDA and SVM, the first machine learning methods to mention include naïve Bayes and nearest neighbours. The former is used for classification only, based on a proper modification of the Bayes formula, while the latter may be used for both classification and regression, based on a distance measure among records (a sort of supervised version of distance-based clustering methods). They will be introduced in Subsection 3.4.1 and 3.4.2 respectively.

Compared to naïve Bayes and nearest neighbours, *classification and regression trees* (CARTs) are definitely more complex. Several algorithms are proposed in literature, more or less common, but the idea stays the same, that is, building a decision tree that can classify or predict a target variable. Given that CARTs are fully non-parametric, they can potentially fit any dataset without error, so they also need pruning techniques to avoid overfitting on out-of-sample data. However, the absence of parameters makes them extremely flexible, and that is why they often outperform parametric methods. We will extensively discuss CARTs in Subsection 3.4.3.
A trade-off between pure parametric methods and pure non-parametric methods is represented by *neural networks*. Their structure is vaguely inspired to the biological neural networks into animal brains. Mathematically, they are defined by underlying activation functions of several parameters that interact with each other. While those functions are predetermined, the parameters need to be trained with a proper optimization algorithm.
Originally, neural network researchers focused on artificial intelligence problems, that is, they tried to replicate the human brain behaviour. However, little by little, the attention moved to other fields, where neural networks are widely used today. They include computer vision, speech recognition, image recognition, social network filtering, medical diagnosis and so on. As a consequence, several structures of neural network have been created to tackle specific tasks. In particular, we must mention *artificial neural networks* (the basic model to classify or predict target variables), *recurrent neural networks* (used to recognize patterns across "time" such as texts and speeches) and *convolutional neural networks* (used to recognize patterns across "space" such as images and videos). Notice that both convolutional and recurrent neural networks are part of a broader family of machine learning methods, called *deep learning*, which focuses on the interpretation of unstructured data. Nonetheless, artificial neural networks are complex enough to solve many actuarial problems, so they will be discussed in detail in Subsection 3.4.4, while other types of neural networks are beyond the scope of this dissertation.
By definition, the most flexible machine learning methods are the so-called *ensembles*, that is, combinations of two or more techniques. Combinations may be totally arbitrary, as long as they are useful to solve specific problems, improving the accuracy and/or reducing the variance of the underlying techniques. However, some more general ensemble methods - those with stronger theoretical foundations - are widely used by data scientists. They include bagging, random forests and boosting, which are typically used to overcome the CART instability by providing an alternative to the traditional pruning. All of them will be presented in Section 4.4
Before concluding this subsection, it is worth pointing out a couple of remarks. First, the distinction between unsupervised and supervised does make sense in most of the cases, but it should not be considered too rigidly.

For instance, CARTs and CART ensembles are often used for data manipulation tasks such as missing value handling or variable selection. In data science, flexibility is everywhere, so any distinction should be taken lightly. Moreover, as the reader can easily imagine, the aforementioned range of methods is far from exhaustive. There exists a huge number of data-specific or task-specific adjustments, adaptations and corrections to the techniques we described so far. This is just another expression of the flexibility offered by data science and machine learning.

1.6.3 Software

In the last years, a number of software houses and developers started delivering their proprietary data science packages. Some of them are rigidly based on predefined algorithms, while others are flexible enough to allow for customizations and enhancements. A comprehensive analysis of these tools is beyond the scope of this dissertation, but a brief introduction of the current trends will be useful to justify our software choices.
Figure 1.11 reports the most used tools in analytics by data scientists according to the KDnuggets surveys in the years 2014-2016 (we included the tools voted in 2016 by 5% respondents at least). Notice that a minor part of the mentioned tools (e.g., Hadoop and Spark) are used for big data management rather than analytics and machine learning.
First, it is worth observing the overall increase in percentages during the period. This is happening in parallel with the general rise in interest on analytics and big data, which leads to the usage of more tools at the same time. Very few tools (e.g., RapidMiner, KNIME, Weka and SAS EM) lost users across the period.
While R strengthens its role as the preferred software among data scientists, Python is rising as the object-oriented programming language for analytics. On the other hand, two classical tools are consistently among the most widely used, that is, SQL and MS Excel. The former is the most common programming language for relational database manipulation, while the latter is the most common spreadsheet-based tool. In addition, it is worth mentioning Tableau, the open-source tool used for data visualization.
In this dissertation, we will use both R and MS Excel. While the former will be used for statistical analysis, the latter will be used for reporting. Although this choice does not allow for the level of flexibility provided by other tools (e.g., Python), we found it to be enough to prepare and manipulate the datasets, launch the algorithms we need for our analyses, and gather the related results.

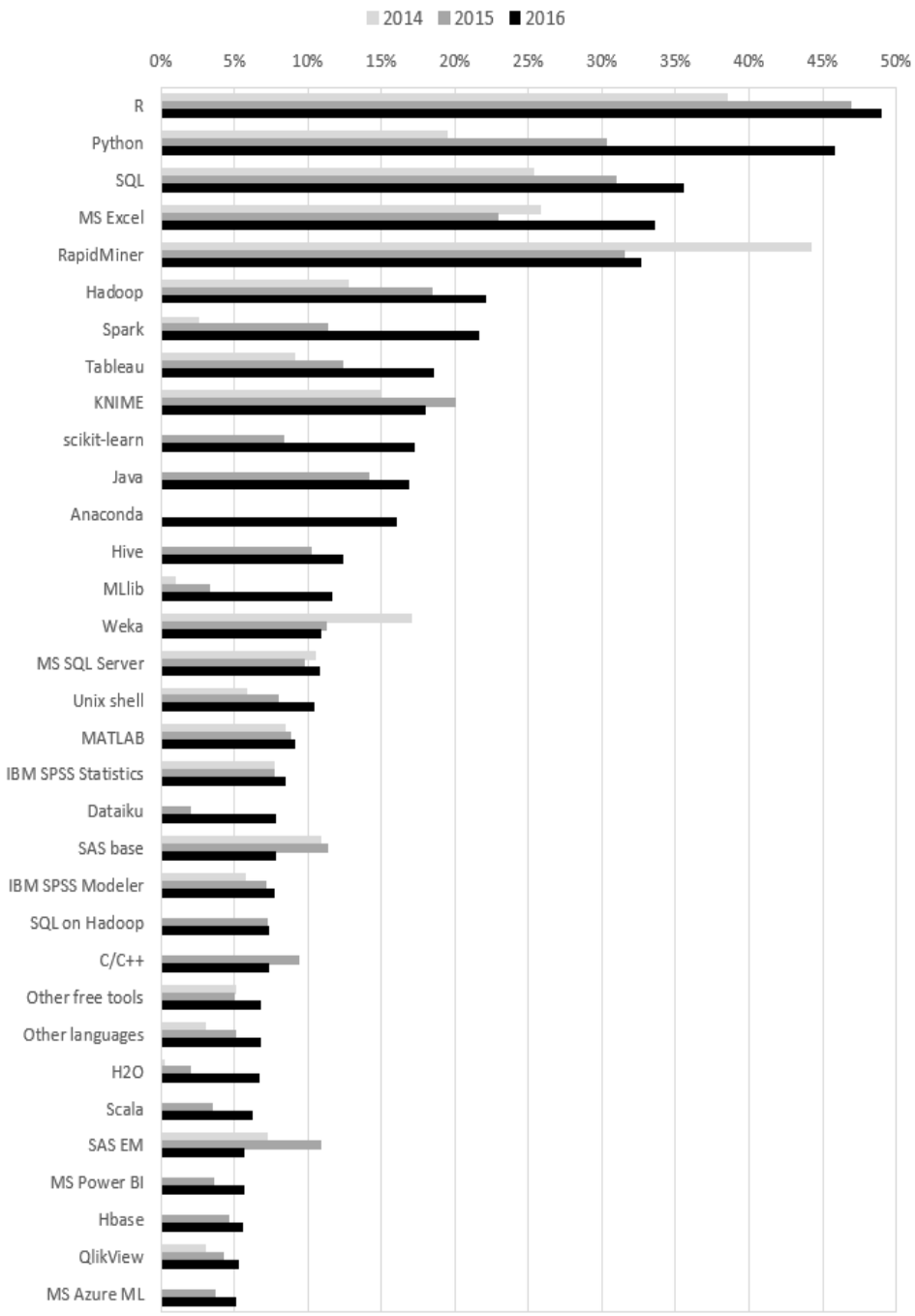


Fig. 1.11: Software poll results in 2014-2016 (see Piatetsky (2016))

Chapter 2

Customer Management using Unsupervised Learning

Customer management is not only a relevant topic in insurance, but also one of the most common and successful applications of data science in the business world. As such, it is worth presenting it as first case study of data science in our industry.

Actuarial context

At a first sight, customer management might not seem a typical actuarial topic. It is instead part of technical assumptions when evaluating in-force portfolio (e.g., renewal and lapse probability) as well as new business (e.g., purchase probability). Setting up actuarial assumptions involves reasonableness, and acceptability is of prime importance. While assumptions such as death rate, disability rate and salary increase have historically been object of many studies, those related to customer management did not catch the same interest. Perhaps, this was partially due to its irrational aspect preventing the usage of analytical models, or the necessity of significant amount of data to extract useful information. Nonetheless, these are exactly the reasons why data science is so useful in predicting customer behaviour.

Chapter overview

In order to imitate the classic structure of data science manuals, this chapter will focus on data handling with unsupervised learning. We will first describe some of the most common tools (i.e., principal component analysis, clustering techniques and associations rules), which will be then applied to manipulate motor insurance data from the French market.
Nonetheless, unsupervised learning represents a preliminary step before the extraction of more useful information from policyholder profiles. We will deal with customer management from two different - though complementary

- perspectives. Firstly, we will use association rules to select the profiles that will most likely buy additional coverages besides the mandatory third-party insurance. Secondly, the dataset will be clustered to predict annual churn rates with logistic regression and get insights into key risk factors of motor insurance renewals.
All in all, the chapter will highlight the importance of unsupervised learning not only as a tool for data handling, but also as a potential performance booster in supervised learning.

2.1 Introduction

Not surprisingly, getting and maintaining clients is a top priority for every company trying to grow its business. In today's highly competitive marketplace, it is extremely challenging for insurance companies to attract and retain customers. Companies take a wide range of strategic actions to get customers to buy and keep their services. The related economic value is widely recognized: however, as active customer management strategies are important for companies to get and retain loyal customers, the ability to correctly determine the drivers of their choices is necessary. This issue actually concerns several markets besides insurance, and it has quickly become one of the most common application fields for data science.
Customer attraction is broadly related to marketing. It is the primary goal of any insurance company, not only for mere revenue reasons, but especially because the insurance business itself can exist - and be profitable - if a portfolio is large enough to allow for risk mutuality.
Companies always try to sell new business by proposing new products either to current customers (i.e., policyholders that are already in portfolio, and own other products) or potential customers: both of them are equally difficult. New customer may already own products of other insurance companies, so it can be extremely hard to pull them away, even with a material reduction in price or a strong increase in quality. At the same time, selling products to current customers may seem easier as they are more accessible, but companies should be able to propose products that meet their actual needs, or even their future ones.
In motor insurance, for example, companies face both the problems. On one hand, third-party coverages are often mandatory in developed markets, so there are new potential customers everywhere, looking for low-cost tariffs rather than specific levels of quality. On the other hand, a number of coverages can be offered together with the basic third-party guarantee, so companies need to detect current customers that are available to customize their products according to their needs.
The analysis of the features of a group of customers to detect potential sale opportunities is called *market basket analysis*, which is a good starting

point. However, as soon as new customers are successfully acquired, a new problem arises: their retention. As reported in Figure 2.1, non-life business in OECD countries is affected by relatively low annual levels of retention, with an average around 72%, corresponding to the *churn rate* of 28%. Even if this is a very general view which does not account for potential differences among branches and company size (in particular, churn rates of the largest insurance companies must be much lower than those of the others), it provides us with a realistic idea of the problem. Whereas the insurance industry has the highest acquisition costs of any industry, it is clear that higher customer retention rates are strong correlated with greater profits. Hence, churn refers to the loss of a customer in general, but it also implies that the profit vanishes.

Customer retention is linked to several business activities such as pricing, marketing and claim management. For instance, an inefficient claim liquidation process will probably lead to lower retention rates. These potential sources of churn should be continuously monitored in order to discover the fundamental drivers of retention, and, as a consequence, try to improve industry processes. However, churn rates are not exclusively caused by inefficient practices. Rather, they depend on the customer profile as well, that is, age, family, job and so on. Such features together with customer historical data may suggest future behaviour. The question is, how much information do the companies have access to? Do they have historical information that can be understood as experimental data? Do they know their customers satisfaction?

The insurance market is a free market where customers may decide which company to become attached to. From the perspective of an insurer, this is even more problematic when it comes with mandatory business such as third-party motor insurance: the company must retain the policyholder as long as he/she decides to change insurer.

Typically, a motor insurance policy lasts for one year and is renewed after that period, if the risk insured still exists. As discussed, this renewal action must be taken by the same policyholder, who has some time to decide if he wants to switch or not to another company. In this regard, we could think about two types of loyalty. According to McKinsey&Company (2012):

> *Loyal policyholders are not a monolithic group of satisfied customers; a subset of loyalists do fit this description, but another subset are identified by the survey as "loyal" in name only. That is, they remain more out of inertia than out of satisfaction. Members of this significant minority of "passive" loyalists can, however, be dislodged and represent significant value hiding in plain sight.*

Beside hidden value, passive loyalists represent a source of hidden risk too. As an example, assume some of them relate to high claim frequencies. The

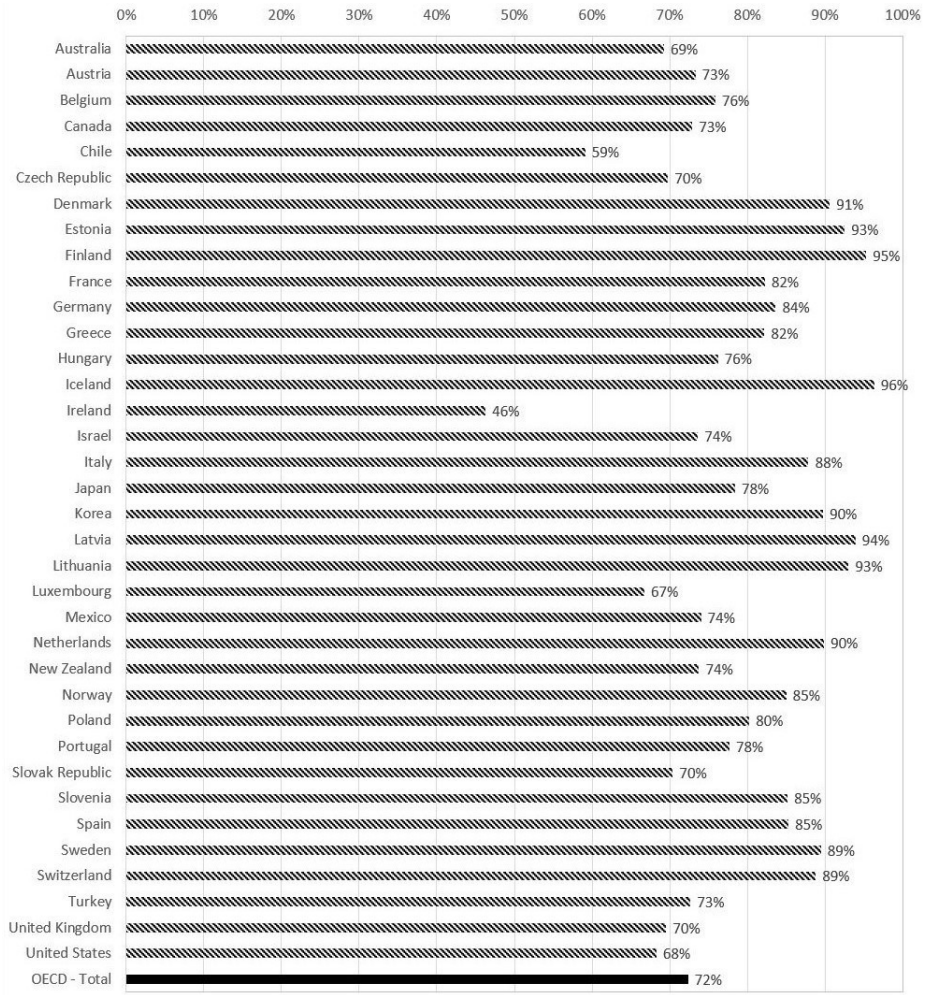


Fig. 2.1: Non-life retention in 2016 (OECD Insurance Indicators (2016))

insurance company will start increasing their premiums, year after year, to cover their claims as well as push them away and indirectly force the churn. However, their inertia will make them stay longer, leading to further losses for the company.
In parallel to customer attraction, customer retention should assume a top role in non-life insurance, not only holding more and more customers: a correct prediction of retention probabilities can also be useful in increasing company profit.
As the first application-based chapter of this dissertation, the next sections will play two roles. First, they should suggest - and, to some extent, prove - the extreme relevance of data preparation in machine learning. This phase will be supported by some unsupervised learning tools introduced in Section 2.3. Such tools are often used as data manipulation tools by data scientists: rather than classifying or predicting, they operate on the dataset as a whole for preparation, reduction or grouping purposes. The second, more practical goal involves market basket analysis using association rules, discussed in Subsection 2.4.3, and churn rate estimation using logistic regression, discussed in Subsection 2.4.4. Beside logistic regression, we could obviously use other machine learning tools for it, but they will be introduced in the next chapters to be used for the related applications only.

2.2 Unsupervised learning in actuarial practice

Unsupervised learning encompasses any tool or algorithm which does not focus on predicting target variables. For instance, it is often used to reduce or group a dataset before further supervised learning analyses. From this perspective, we can see unsupervised learning as a useful (if not even necessary), preliminary step to support complex data manipulation. Obviously, it may be also used for standalone analysis (e.g., market basket analysis), although practical applications are less evident to actuaries.
Actually, the actuarial field is full of opportunities to build supervised learning models and predict target variables more accurately (two of those examples will be tackled in Chapter 3 and 4). As a consequence, according to the actuarial tradition, a model should always provide an outcome, an estimation. However, this is not the case with unsupervised learning, where the distinction between input and output is less clear. Nonetheless, there are already few common applications in the actuarial field, so it is worth outlining a short overview on them.
Since years, financial market experts - including actuaries - build simulation methods to forecast interest rate curves in the future. One of the most common and intuitive approach is Monte Carlo simulation: first, we define a stochastic model for the interest rate dynamics, and then simulate it through a large number of random numbers (this is essentially what we will do in

	PC1	PC2	PC3	PC4	PC5
Germany (€)	71.5%	84.8%	92.9%	96.7%	98.4%
Japan (¥)	69.7%	90.7%	95.6%	97.7%	99.2%
United States ($)	84.4%	90.9%	94.4%	97.3%	98.9%

Fig. 2.2: Cumulative explained variance of the first five principal components on the German, Japanese and US interest rate curves in 1995 (Jamshidian et al. (1997))

Chapter 4). Now, assume that each random number can equal one of n predetermined values, while the stochastic model depends on k parameters to describe the yield curve. Therefore, if we need the full distribution of the future curves, we will have to simulate n^k curves. For instance, if $n = 5$ and $k = 10$, we will get almost ten millions different curves! Even if Monte Carlo never requires all the permutations, this example should show that the more the parameters, the heavier the method and the more the simulations needed to guarantee a reasonable accuracy.
On the other hand, many studies (e.g., Kahn (1989), Gulrajani et al. (1995), Rebonato (1998), etc.) empirically proved that two or three factors can be sufficient to explain the change in interest rates. For example, Figure 2.2 reports the PCA results for the yield curves in Germany, Japan and United States in 1995 (see Jamshidian et al. (1997)). We can immediately notice that the first three components explain a large amount of information, and for the Japanese and the US curves, two components are sufficient to explain more than 90% variability. This is the reason why many stochastic models are defined by few parameters. However, predetermining a fixed number of degrees of freedom is still a strong limitation.
PCA represents one of the most common methods to extract the fundamental components of yield curves. It provides us with the flexibility of choosing the most proper number of components depending on our own accuracy tolerance. This is not only the topic of a number of more or less recent studies (e.g., Charpentier et al. (2010), Liu (2010), Lord et al. (2007) and Schmidt (2011) as well as the aforecited Kahn (1989), Gulrajani et al. (1995) and Rebonato (1998)), but also an approach widely accepted by regulatory authorities worldwide. For instance, EIOPA used PCA to calibrate the interest rate shocks in the standard formula (see EIOPA (2014)):

> *The calibration of the interest rate shocks in the standard formula are based on the relative changes of the term structure of interest rates using the following 4 datasets: EUR government zero coupon term structures (1997 to 2009), GBP government zero coupon term structures (1979 to 2009), and both Euro and GBP LIBOR/swap rates (1997 to 2009). For each of the four individual datasets, stress factors were assessed through a Principal Component Analysis (PCA), according to their maturity. PCA*

> *is a tractable and easy- to- implement method for extracting market risk factors. For each maturity, the mean of the results in the four datasets was taken as a single stress factor.*

More recently, the new European Directive on Packaged Retail Investment and Insurance Products (PRIIPs) regulation explicitly requires PCA for the simulation of the fund performance in the market risk assessment (see PRIIP Regulation (2016), paragraph 23):

> *For curves, a principal component analysis (PCA) shall be performed to ensure that the simulation of the movements of each point on the curve over a long period results in a consistent curve.*

Among unsupervised techniques, clustering is even more common than PCA in actuarial practice and research. For instance, Pelessoni et al. (1998), Sanche et al. (2006), Guo (2003) and Wüthrich (2016a) outline various clustering applications in insurance, especially in general insurance, because this is one of the fields characterized by the greatest variety and variability.
In fact, cluster analysis is often used to partition policyholders based on the risk they represent: this is usually called *ratemaking*. There are a number of reasons to perform ratemaking on an insurance portfolio before further analyses. First of all, each cluster represents a specific group of policyholders sharing common features: if such features are determined, the company may customize its management and strategy according to them. Second, features within each cluster are less volatile than those within the whole portfolio, leading to more robust pricing and reserving. A last reason relates to the reduction of levels in rating factors, which are often very numerous in general insurance: in other words, each cluster could suggest specific aggregations of such levels in order to reduce dimensionality.
As an example, territorial clustering is already part of the ratemaking process in motor insurance. Two relevant contributions on territorial clustering are represented by Yao (2008) (further discussed in Parodi (2009) and Frees et al. (2014), among others) and Jennings (2008), published in the same year (2008) by the Casualty Actuarial Society. In Yao (2008), the author applies a number of clustering methods to compare the different partitions returned, and improve the basic GLM-based pricing (Figure 2.3 shows one of the partition returned by the K-means method). The fundamental idea of Jennings (2008) is the same, but it focuses on US data.
As already discussed, unsupervised learning applications are less common in the actuarial field. However, besides widespread instances such as interest rate modelling with PCA and territorial ratemaking with clustering, other applications that are already common in other industries (e.g., sentiment analysis, text recognition, etc.) could start playing a role in the actuarial practice very soon.

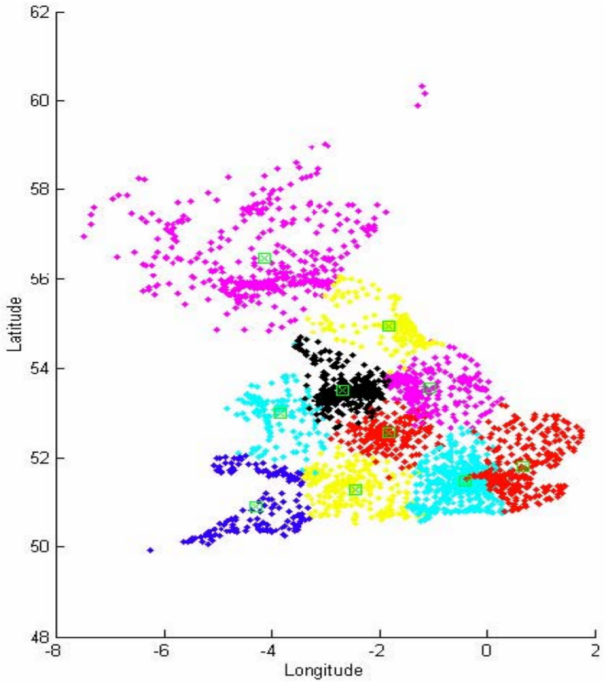


Fig. 2.3: Example of territorial clustering using the K-means method on a UK motor insurance portfolio (Yao (2008))

2.3 Fundamental unsupervised learning tools

Recalling what was explained in Section 1.1, unsupervised learning encompasses any type of tool which does not need a formal target variable. This category encompasses, among others, association rules, PCA and clustering techniques. They are all used for very different purposes, but none of them predicts or classifies records.
In this section, we will introduce the aforementioned tools, explaining the mechanisms as well as the purposes. After that, we will be ready to use them as supporting tools for market basket analysis and churn rate estimation on our motor insurance dataset.

2.3.1 Association rules

Association rule learning is a basic machine learning method to discover strong rules between variables in large datasets. It has been introduced in Agrawal et al. (1993) for the first time as a market basket analysis tool: the authors look for stable and robust regularities between products bought by single customers. In particular, they want to answer to the following question:

> Given that the customer bought a specific range of products,

> which is the probability that he/she will further buy one specific item?

which is quite similar to what will be addressed in the application of this chapter.
First of all, an association rule is any if-then statement between explanatory variables in a database: if A (the *antecedent*) occurs, then B (the *consequent*) occurs as well. Generally, A and B may represent a combination of features rather than one single feature. However, we will always assume that for A only, not for B.
Of course, some rules will be strong, while others will be weak. The most intuitive measures of the robustness of one specific rule is its *a priori* likelihood. This is the reason why association rules are often generated by the so-called *apriori algorithm*, originally proposed in Agrawal et al. (1993). Let's consider a dataset with categorical variables $x_1, \dots, x_n$, where each of them is defined by a given number of categories, say $k_1, \dots, k_n$. For instance, the categories of x_i will be denoted by $x_{i,1}, \dots, x_{i,k_i}$.
In the first step, we pick a specific consequent, say $x_{n,1}$, and then start keeping track of all the records with that category. The antecedent(s) may vary among all the other variables, so we count all the records including the aforementioned consequent and any combination of the remaining features. For instance, we count the records including $x_{n,1}$ as well as $x_{1,1}, \dots, x_{n-1,1}$, so their proportion on the whole dataset defines the *support* of the following rule:

$$x_{1,1}, \dots, x_{n-1,1} \Longrightarrow x_{n,1}. \tag{2.1}$$

More generally, the support can be defined as follows:

$$\text{support} := P(\text{antecedent},\ \text{consequent}). \tag{2.2}$$

As such, the greater the support, the stronger the association rule, so that is the first discriminant to detect significant rules. By setting a minimum support level, we can exclude all the rules whose support is lower. This is not only important as a primary way to pick rules, but also a way to reduce computational complexity. Indeed, if we really had to consider each and every possible rule, it would lead to an extremely high number of operations. In particular, if $k_i \equiv k$ for each i, the number of single-consequent rules is

$$k^n \sum_{i=1}^{n-1} \binom{n-1}{i} \tag{2.3}$$

and, as an example, if the dataset has only five binary variables ($n = 5$ and $k = 2$), the possible association rules are 480!
However, the support threshold can partially fix the problem. First, notice that the support depends on the features appearing in that rule regardless

of their role, either antecedent or consequent. For instance, the support of the rule in (2.1) is the same as the support of

$$x_{2,1}, \ldots, x_{n,1} \Longrightarrow x_{1,1}. \tag{2.4}$$

As a consequence, if the support of (2.1) is below the support threshold, the support of (2.4) and that of any other *affine* rule will be below it. Therefore, we can immediately exclude all those rules from the analysis without even checking them. This means that the number of rules in (2.3) leads to the evaluation of k^n supports only. In the dataset of five binary variables, it means 32 evaluations rather than 480.
Moreover, the number of significant rules is further reduced by another remark. Assume that the support of

$$x_{1,1} \Longrightarrow x_{n,1} \tag{2.5}$$

is below the threshold. Then we can immediately exclude all the rules including $x_{1,1}$ and $x_{n,1}$ such as those in (2.1) and (2.4).
With that in mind, the algorithm repeats the search of association rules varying the consequent features: first $x_{n,1}$, then $x_{n,2}$ and so on. When all the significant x_n-consequent rules are detected, the algorithm sets a new consequent, say x_{n-1}, and restarts the search for all of its categories. When all the variables have been parsed, the algorithm stops and returns a list of rules, each with a number of antecedents and one single consequent. Nonetheless, if we knew that only some variables are to be considered as consequent, then we can limit the algorithm to them in order to make it even faster by avoiding useless operations.
The support threshold provides a criterium to get the list of rules "supported" by a sufficient proportion of records. However, not all of them are actually strong. Since any rule is defined by antecedent and consequent, we are interested in the proportion of records showing the consequent feature *given that they show the antecedent feature(s).* This is called *confidence* of the rule:

$$\text{confidence} := P(\text{antecedent} \mid \text{consequent}) = \frac{P(\text{antecedent},\ \text{consequent})}{P(\text{antecedent})} \tag{2.6}$$

that is the conditional probability of the consequent given the antecedent. If the support level is around 80%, but the confidence level is relatively low, say 50%, we can still say that the rule is statistically significant since it considers 80% of the available records, but the antecedent implies the consequent only half the time. In other words, we cannot state that the former actually implies the latter, and the rule turns out to be very weak. This is not the case if the confidence reaches higher values such as 90%: at those levels, the rule can be considered quite strong.

However, the confidence level is not necessarily the best measure of a rule's strength. To make it evident, let's consider the extreme example of independence between antecedent and consequent. In this case, the confidence in (2.6) is reduced to $P(\text{consequent})$, which may be quite high if the consequent feature is frequent although there is no association by definition.
To fix this issue, we can use $P(\text{consequent})$ as a *benchmark confidence*, that is, the minimum acceptable confidence level for a rule should be $P(\text{consequent})$.
In other terms, we can define a new measure, the *lift ratio*, which encompasses both the confidence and the benchmark confidence:

$$\text{lift ratio} := \frac{\text{confidence}}{\text{benchmark confidence}} = \frac{P(\text{antecedent, consequent})}{P(\text{antecedent})P(\text{consequent})} \quad (2.7)$$

which indicates some usefulness to the rule if it is greater than 1, although it is not enough to state that the rule is actually strong. The greater the lift ratio, the stronger the association.

2.3.2 Principal component analysis (PCA)

PCA represents not only one of the most popular multivariate statistical techniques, but also the oldest one. Indeed, its origin can be traced back to Pearson (1901), if not even Cauchy (1829) or Jordan (1874). However, the modern formalization has been introduced in Hotelling (1933) together with the concept of "principal component" as a dimension reduction tool.
Generally speaking, information is affected by some level of correlation among variables. These dependencies make data partially redundant, so there should be room for variable reduction. That is the goal of PCA: exploiting this room to extract new, independent variables - the principal components - to replace the original ones. The number of principal components will always be the same as the number of original variables, but we may choose a subset of them in order to reduce the size of the dataset while keeping a reasonable amount of information. Therefore, the greater the (linear) correlations, the greater the PCA benefit. If the data does not show any correlation, PCA will return the original dataset itself.
First of all, let N be the number of data points and n the number of variables, which should be numerical and standardized. Let's represent such a data matrix with $\boldsymbol{X}$ and the related covariance matrix with $\boldsymbol{\Sigma}_X$. Given that any covariance matrix is positive semi-definite (i.e., there exists a matrix $\boldsymbol{A}$ such that $\boldsymbol{\Sigma}_X = \boldsymbol{A}\boldsymbol{A}^T$), $\boldsymbol{\Sigma}_X$ can be "eigen-decomposed", as we will explain.
Recall that an *eigenvector* of $\boldsymbol{\Sigma}_X$ is defined as the vector $\boldsymbol{u}$ that satisfies the following equation:

$$\boldsymbol{\Sigma}_X \boldsymbol{u} = \lambda \boldsymbol{u} \quad (2.8)$$

for some λ, which is called the *eigenvalue* associated to $\boldsymbol{u}$. Notice that if $\boldsymbol{u}$ is an eigenvector, then any vector $a\boldsymbol{u}$, for any constant a, is an eigenvector

as well. In order to avoid those sort of redundancies, eigenvectors are always normalized in practice, that is, transformed so that their norm is 1 (i.e., $\boldsymbol{u}^T\boldsymbol{u} = 1$). In this way, it can be proven that $\boldsymbol{\Sigma}_X$ admits exactly n eigenvectors and n related eigenvalues. Moreover, if we define U as the matrix of such (column) eigenvectors and Λ as the diagonal matrix of such eigenvalues, we can use (2.8) to write

$$\boldsymbol{\Sigma}_X\boldsymbol{U} = \boldsymbol{\Lambda}\boldsymbol{U} \tag{2.9}$$

or

$$\boldsymbol{\Sigma}_X = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1} \tag{2.10}$$

which represents the *eigen-decomposition* of $\boldsymbol{\Sigma}_X$.
Positive semi-definite matrices like $\boldsymbol{\Sigma}_X$ not only admit such a decomposition, but the eigenvalues are all distinct and nonnegative, implying the orthogonality of the eigenvectors. Then, the eigenvector matrix $\boldsymbol{U}$ is orthogonal as well, that is

$$\boldsymbol{U}^{-1} = \boldsymbol{U}^T \tag{2.11}$$

and (2.10) becomes

$$\boldsymbol{\Sigma}_X = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \tag{2.12}$$

which represents the *diagonalization* of $\boldsymbol{\Sigma}_X$. In practice, (2.12) expresses the transformation of the original covariance matrix $\boldsymbol{\Sigma}_X$ to a new covariance matrix $\boldsymbol{\Lambda}$ which is diagonal, that is, includes variances only. In fact, such a transformation condensed the information on the diagonal of the covariance matrix, moving it from the correlation components.
So far we converted the covariance matrix to a variance matrix, but that is not enough, because we should transform the dataset itself accordingly. In practice, we need to transform the data points so that their new covariance matrix is diagonal. More specifically, let's find the matrix $\boldsymbol{P}$ such that the covariance matrix of the new dataset $\boldsymbol{Y} := \boldsymbol{P}\boldsymbol{X}$ equals $\boldsymbol{\Lambda}$. Notice that, if we use (2.12) and define $\boldsymbol{P} := \boldsymbol{U}^T$, then the covariance matrix of the new dataset is

$$\begin{aligned}
\boldsymbol{\Sigma}_Y &= \frac{1}{N}\boldsymbol{Y}\boldsymbol{Y}^T = \frac{1}{N}(\boldsymbol{P}\boldsymbol{X})(\boldsymbol{P}\boldsymbol{X})^T = \frac{1}{N}\boldsymbol{P}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{P}^T = \\
&= \boldsymbol{P}\Big(\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^T\Big)\boldsymbol{P}^T = \boldsymbol{P}\boldsymbol{\Sigma}_X\boldsymbol{P}^T = \boldsymbol{P}(\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T)\boldsymbol{P}^T = \\
&= (\boldsymbol{P}\boldsymbol{U})\boldsymbol{\Lambda}(\boldsymbol{U}^T\boldsymbol{P}^T) = (\boldsymbol{U}^T\boldsymbol{U})\boldsymbol{\Lambda}(\boldsymbol{U}^T\boldsymbol{U}) = \\
&= (\boldsymbol{U}^{-1}\boldsymbol{U})\boldsymbol{\Lambda}(\boldsymbol{U}^{-1}\boldsymbol{U}) = \\
&= \boldsymbol{\Lambda}
\end{aligned} \tag{2.13}$$

which is exactly our aforementioned goal.
All in all, the matrix that transforms the data points is the transpose of the matrix that zeroised the correlation components in the original covariance matrix. Once we have built the matrix $\boldsymbol{U}$ by using the eigenvectors of $\boldsymbol{\Sigma}_X$,

the new dataset is easily defined by $\boldsymbol{U}^T\boldsymbol{X}$, whose (co)variance matrix is $\boldsymbol{\Lambda}$. As a consequence, each eigenvalue on the diagonal of $\boldsymbol{\Lambda}$ equals the variance explained by the related principal component, while there is no information brought by correlations since the principal components are independent by construction.
The final step is straightforward: we only need to pick those principal components that explain most of the variance altogether (e.g., more than 80% or 90%, depending on the specific application). In this way, the new dataset still inherits most of the original information, but it is condensed in less, independent variables.
However, we should always take into account two fundamental drawbacks of PCA. First, the basic algorithm described so far extracts principal components based on the linear dependencies among the original variables, while any nonlinear relationship is completely ignored. Secondly, the principal components turn out to be a linear combination of the original variables, and, as such, they lose any practical interpretation. Nonetheless, several enhancements to the standard PCA have been proposed, for instance, the *singular-value decomposition* (SVD), which implements a generalization of the eigen-decomposition to directly transforms the dataset regardless of its covariance matrix.

2.3.3 Cluster analysis

While PCA is a tool to group variables (the "columns"), cluster analysis is a tool to group records (the "rows"). Obviously, records are grouped according to their similarities: similar records are more likely to be grouped than different record. Therefore, any clustering method is first based on the distance measure used to evaluate how close two records are. The most popular choice is represented by the *Euclidean distance*: given the vectors of features for two different records, say $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{in})$ and $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jn})$, the Euclidean distance between them is defined as follows:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) := \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{x}_i - \boldsymbol{x}_j)} = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}, \qquad \forall i, j. \quad (2.14)$$

Notice that the Euclidean distance is highly influenced by the scale of each variable, so we should always cluster a previously standardized dataset, otherwise we may obtain very flawed results.
Although that is the most widely used distance, it has some drawbacks. For instance, it is very sensitive to outliers: we could remove them or use a more robust distance such as the *Manhattan distance*:

$$d_M(\boldsymbol{x}_i, \boldsymbol{x}_j) := |\boldsymbol{x}_i - \boldsymbol{x}_j|^T\mathbf{1} = \sum_{k=1}^{n}|x_{ik} - x_{jk}|, \qquad \forall i, j. \quad (2.15)$$

where $\mathbf{1}$ denotes the n-dimensional vector $(1, \ldots, 1)$. The Euclidean distance also ignores correlations among variables, and it turns out to be a relevant problem when some of them are strongly correlated. That is the reason why a correlation-based distance may be more convenient. For example, if ρ_{ij} denotes the correlation coefficient of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, then we can simply define the distance $d_\rho(\boldsymbol{x}_i, \boldsymbol{x}_j) := 1 - \rho_{ij}^2$. However, given the covariance matrix Σ, there is a more rigorous way to define a correlation-based distance:

$$d_\Sigma(\boldsymbol{x}_i, \boldsymbol{x}_j) := \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \Sigma^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j)}, \qquad \forall i, j \tag{2.16}$$

which is called *Mahalanobis distance*.
Beside the distance between records, cluster analysis requires the choice of a distance measure between clusters, which are instead group of records. Given two distinct clusters $\boldsymbol{X}$ and $\boldsymbol{Y}$ encompassing the records $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p$ respectively, we can define the following distances between them:

$$\text{single linkage} := \min_{i,j} d(\boldsymbol{x}_i, \boldsymbol{y}_j) \tag{2.17}$$

$$\text{complete linkage} := \max_{i,j} d(\boldsymbol{x}_i, \boldsymbol{y}_j) \tag{2.18}$$

$$\text{average linkage} := \frac{1}{mp} \sum_{i,j} d(\boldsymbol{x}_i, \boldsymbol{y}_j) \tag{2.19}$$

$$\text{centroid linkage} := d\left(\frac{1}{m} \sum_{k=1}^{m} x_{ik}, \frac{1}{p} \sum_{k=1}^{p} y_{jk} \right) \tag{2.20}$$

where we use the Euclidean distance d defined in 2.14. Using distance measures, clustering algorithms can properly group records and merge clusters. Moreover, there is a further approach called *Ward's method* that is not really based on the concept of distance. Instead, it takes advantage of the decomposition of the total variance in a dataset between the *within variance* (i.e., the variance "within" each single cluster) and the *between variance* (i.e., the variance "between" different clusters). Such a decomposition can be expressed as follows:

$$\sum_{j=1}^{K} \sum_{i=1}^{N_j} (x_{ij} - \overline{x})^2 = \underbrace{\sum_{j=1}^{K} \sum_{i=1}^{N_j} (x_{ij} - \overline{x}_j)^2}_{\text{within variance}} + \underbrace{\sum_{j=1}^{K} N_j (\overline{x}_j - \overline{x})^2}_{\text{between variance}} . \tag{2.21}$$

where K is the number of clusters, N_j the number of records in the j^{th} cluster, $\overline{x}_j$ the average over the j^{th} cluster and $\overline{x}$ the average over the whole dataset. In fact, while distance-based clustering generates clusters by minimizing distances among records, the Ward's method generates clusters by minimizing variances within clusters. Notice that, since the total variance

is constant by definition, then the smaller the within variances, the greater the between variances. This is reasonable: if little information is exchanged among records within clusters, most of it will be necessarily exchanged outside, that is, between clusters.

Once we have chosen the measure or the rule to group records, we can choose the clustering method. Generally speaking, data scientists distinguish between two fundamental types of clustering: *hierarchical clustering* and *non-hierarchical clustering*. In this subsection, we will introduce the two most common algorithms for both.

Hierarchical clustering is especially known in its agglomerative version, merging close clusters together (hierarchical clustering can be divisive as well, separating distant clusters instead, but it is less common). The algorithm starts with N trivial clusters, one for each data point. In the first step, all the distances among clusters are calculated, and the two closest clusters are merged (in the case of the Ward's method, the clusters merged are those leading to the smallest increase in within variance). The second step is analogous: the distances are recalculated, and the two closest clusters are merged (once again, the Ward's method merges the clusters trying to keep the within variance as small as possible). Going on with further steps, the number of clusters will progressively decrease until all the records are grouped under one single cluster. Then the algorithm stops.

As we can easily figure out, the process creates a cluster hierarchy, starting from the smallest clusters (i.e., the records themselves) to the biggest ones, and finally the whole dataset. As such, we can easily keep track of the clusters generated at each step. That can be done with a *dendrogram*, a treelike diagram that illustrates the whole cluster hierarchy. In Figure 2.4, we reported an example of it. Each vertical segment represents a cluster; in particular, the short segments at the very bottom represents the single records, that is, the initial clusters. The length of those segments reflects the distance between the clusters. Therefore, the horizontal segments denote the merge of two clusters in the related step. Heuristically speaking, we can expect that any dendrogram tends to flatten as the number of clusters increase, in the sense that the distances between the larger clusters on the top side will be probably greater than the distances between the smaller clusters on the bottom side.

Furthermore, dendrograms provide a useful view to select clusters. Any horizontal line on the graph represents a specific choice. In Figure 2.4, for example, the horizontal red line leads to the selection of the four clusters inside the red squares.

In its simplicity, hierarchical clustering can be seen as the most natural way to generate clusters: the closer the clusters, the earlier their aggregation. However, the major drawback of this approach is the computation of $\binom{N}{2}$ distances in the first step, which are also updated at each subsequent step. For instance, if the first two records are aggregated at the end of the first

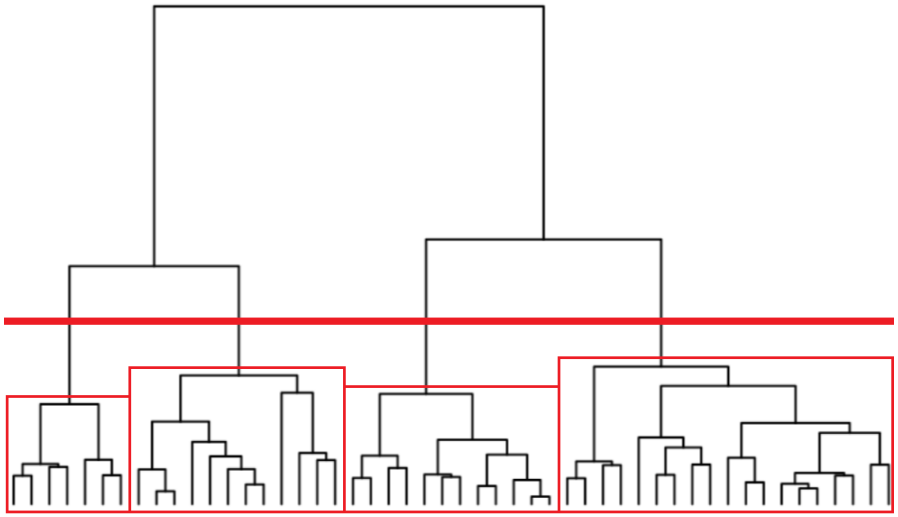


Fig. 2.4: Example of dendrogram

step, then the algorithm will calculate the new $N-2$ distances for the second step. Analogously, if the third record is also aggregated at the end of the second step, then the algorithm will calculate the new $N-3$ distances for the third step and so on. All in all, the process requires $\binom{N}{2} + \frac{(N-2)(N-1)}{2}$ operations at least, which may be very expensive for large datasets. That is especially true since $\binom{N}{2}$ distances need to be allocated in the CPU: this is often infeasible.

For such reasons, non-hierarchical clustering might be a preferred approach, because it does not pose computational issues. In fact, it does not build any full hierarchy over the dataset, so much less information needs to be allocated.

The most common non-hierarchical method is the *K-means algorithm*. First, the algorithm needs some initial conditions, that is, an integer K denoting the number of clusters to generate, and a set of K records representing the initial clusters (and the related centroids). In the first step, each record is assigned to the closest initial cluster, in order to generate a first set of K clusters, and the centroids are recalculated. In the second step, each record is reassigned to the closes cluster among those coming from the previous step, and the centroids are recalculated once again and so on. After a number of steps, the centroids will stabilize and records will no longer move. Notice that the algorithm calculates $(N-K)K$ distances at each step, but it does not need to keep them in memory. This is what makes it much cheaper than hierarchical clustering. Of course, without hierarchy the dendrogram cannot be plotted.

However, the gain in computational complexity reflects in the loss in result

stability. Indeed, the choice of the initial clusters does impact the outcome of the algorithm. Lucky choices may output better results as well as unlucky choices may output worse results. Sometimes there could be strong reasons to choose a specific set of initial clusters, but often this is not the case and the algorithm should be run over a number of different sets of initial condition to individuate stable clusters. This is not only useful to return stable partitions, but also to choose a reasonable K. For example, if we observe a major drop down in dispersion at one specific K, this could be a good reason to choose that K.
Nonetheless, several studies such as Arai et al. (2007), Chen et al. (2005) and Liu et al. (2017) propose some combinations of hierarchical and non-hierarchical algorithms to fix or avoid the problem of setting initial conditions for the K-means algorithm. The resulting algorithm is generally called *hierarchical k-means clustering*. Its simplest and most intuitive form comes out when the centroids returned by the hierarchical method are used as initial conditions of the K-means algorithm (this is the technique we will use to estimate the churn rate later on). Even if we still need to run the hierarchical algorithm, this is only needed one single time to extract reasonable centroids, whereas its clusters and the whole hierarchy will be ignored. After that, indeed, we will use the K-means algorithm only.
All in all, hierarchical clustering is computationally expensive, but quite robust, while non-hierarchical clustering is computationally cheap, but more volatile. Nonetheless, beside computational cost and stability, cluster analysis should also guarantee interpretability. In other words, cluster analysis reaches its goal when the clusters are meaningful, somewhat in line with our expectations. When this is possible, then they can be labelled and figured out qualitatively, not only numerically.

2.4 An application to French motor insurance data

The goal of our analysis is to predict customer behaviour in a given motor insurance portfolio, and explain this by examining which features affect the purchase and retention of the product. We will use the unsupervised tools described in Section 2.3 to support data preparation and analysis. After that, the churn rate will be predicted by a simple logistic regression (further supervised machine learning tools will be introduced and used in Chapter 3 and 4).

2.4.1 Data

The data comes from an R package containing a number of datasets for actuarial applications (see Dutang et al. (2016)). In Dutang et al. (2016), the whole database is called *French claims for private motor* (`fremotorclaim`),

although it encompasses, among others, three different datasets with the following description:

> `fremotor1freq`, `fremotor1sev`, `fremotor1prem` *are three datasets from the same database for a private motor portfolio observed between January 2003 and June 2004, respectively the claim frequency database, the claim severity database and the premium database.*
> *The dataset* `fremotor1prem` *consists of 50.710 records with explanatory variables for policies (possibly with multiple vehicles insured under the same policy number).*
> *The dataset* `fremotor1freq` *consists of 19.928 records of claim numbers (by policy) between January 2003 and June 2004.*
> *The dataset* `fremotor1sev` *consists of 18.057 records of claim amount, their occurrence date, the corresponding guarantee, between January 2003 and June 2004.*

Our starting point is represented by `fremotor1prem`, which contains 31 fields:

- `IDpol` for the policy ID (used to link with the claims in `fremotor1sev`)
- `DrivAge` for the driver age, in years
- `DrivGender` for the gender
- `MaritalStatus` for the marital status
- `BonusMalus` for the bonus/malus, between 50 and 350 (lower than 100 means bonus, grater than 100 means malus in France)
- `LicenceNb` for the licence number (at least one)
- `VehNb` for the power of the car (ordered categorical)
- `PayFreq` for the payment frequency
- `JobCode` for the job code
- `VehAge` for the vehicle age, in years
- `VehClass` for the vehicle class
- `VehPower` for the vehicle class from least powerful (P2) to most powerful car (P15)
- `VehGas` for the car gas (diesel or regular)
- `VehUsage` for the vehicle usage

- `Garage` for the type of garage
- `Area` for the area code
- `Region` for the policy regions in France (based on a standard French classification)
- `Channel` for the channel distribution code
- `Marketing` for the marketing code
- `PremWindscreen` for the premium of windscreen guarantee
- `PremDamAll` for the premium of damage all-accident guarantee
- `PremFire` for the premium of fire guarantee
- `PremAcc1` for the premium of type-1 accident guarantee
- `PremAcc2` for the premium of type-2 accident guarantee
- `PremLegal` for the premium of legal protection guarantee.
- `PremTPLM` for the premium of mandatory third-party liability guarantee
- `PremTPLV` for the premium of voluntary third-party liability guarantee
- `PremServ` for the premium of service guarantee
- `PremTheft` for the premium of theft guarantee
- `PremTot` for the total premium
- `Year` for the calendar year.

Moreover, the dataset `fremotor1sev` contains 5 fields:

- `OccurDate` for the occurrence date
- `Payment` for the amount of money paid
- `IDpol` for the policy ID
- `IDclaim` for the claim ID
- `Guarantee` for the corresponding guarantee of the claim

and the dataset `fremotor1freq` contains 3 fields:

- `IDpol` for the policy ID
- `ClaimNb2003` for the claim number in 2003

- `ClaimNb2004` for the claim number in 2004.

Notice that `fremotor1sev` implicitly includes the data of `fremotor1freq`, so we can ignore the latter as a first step. Then, the factor variables `Garage` and `Marketing` explain exactly the same information, that is, there is no difference between their categories, so we can delete one of them, say `Marketing`. For the same reason, we can delete `PremTot`, which is the sum of the single premium components.
One of our goals is the estimation of the churn rate in year 2004, that is, the probability that a policyholder in portfolio as at year 2003 will not renew the year after. To do that, we need to separate the records per `Year`. In particular, we will use the 2003 portfolio as our current portfolio and perform our analysis on that, while the only information needed from the 2004 portfolio is the indication about the presence/absence of the records that existed in 2003. This information will be represented by the new binary variable `Churn`. Obviously, the field `Year` is constantly 2003 now, so it can be deleted.
Remember that data are available until June 2014, that is, there is a chance that we are overestimating the churn rate. To address this potential issue and avoid excessive flaws in our analysis, various checks were necessary. For instance, if policyholders enter the dataset as soon as they pay their premium, we should expect the renewal to be wrongly driven by `PayFreq`: indeed, about half of the policyholders with annual premium frequency should pay the premium and enter the dataset after June 2014. However, this is not the case, and more generally payment frequency seems to be completely unrelated to the renewal (in particular, it will not be retained by the stepwise regression in Subsection 2.4.4). All in all, we are aware of the potential flaws, but the data as well as the final results do not seem to be strongly impacted.
As explained in the database description from Dutang et al. (2016) reported at the beginning of this subsection, `fremotor1prem` includes duplicated policy IDs. It is important to check them to determine whether they refer to one of the followings:

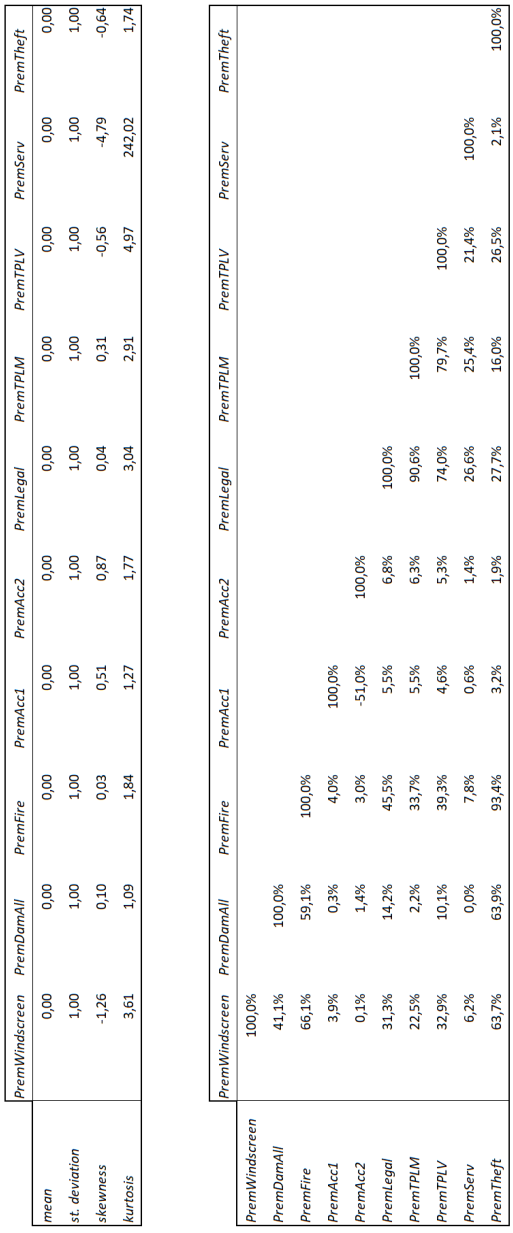
1. one vehicle insured by one policyholder
2. more vehicles insured by one policyholder
3. one vehicle insured by more policyholders
4. more vehicles insured by more policyholders.

Unfortunately, we do not know the real reasons of such duplicates. For example, the case 1. might refer to a change in premium amount, while the case 3. might refer to the same family or firm. In any case, we need to keep this type of information as much as we can, and we do it by defining two new

fields, that is, `OwnedVehiclesAssured` and `FamilyMembersAssured`, both of them equal to 1 as default.
Regarding the duplicates of the first type, we simply keep the last record in chronological order, which is inferable from some features of the duplicates such as driver age or vehicle age (the other subsequent duplicates are deleted).
Regarding the duplicates of the second type, we can detect them by using the vehicle features at our disposal to determine whether the duplicates relate to different vehicles: if so, we keep both the records, and set their `OwnedVehiclesAssured` to 2 (or more, in case of triplicates, quadruplicates, etc.).
Regarding the duplicates of the third type, we can detect them by using the policyholder features at our disposal to determine whether the duplicates relate to different policyholders: if so, we keep both the records, and set their `FamilyMembersAssured` to 2 (or more, in case of triplicates, quadruplicates, etc.).
Finally, regarding the duplicates of the fourth type, we can detect them by using both the vehicle features and the policyholder features at the same time, keeping all the duplicates, and setting `OwnedVehiclesAssured` and `FamilyMembersAssured` accordingly.
As a conclusion, we delete few records relating to categories that are very infrequent in the dataset, that is, those with `LicenceNb` equal to 6 or 7, those with `VehNb` equal to 6 or 21, those with `JobCode` equal to Retailer, those with `Area` equal to A12, those with `OwnedVehiclesAssured` equal to 4, and those with `FamilyMembersAssured` equal to 3. That should make results stabler.

2.4.2 Data preparation with machine learning

In this subsection, we will complete the data preparation started in Subsection 2.4.1. More specifically, we will first perform dimension reduction on the premium amount fields, and then the claims of the dataset `fremotor1sev` will be assigned to the records of `fremotor1prem`.
As explained in Subsection 2.3.2, PCA is one of the most common techniques for dimension reduction. First, we take the natural logarithm of the premium variables (`PremWindscreen`, `PremDamAll`, `PremFire`, `PremAcc1`, `PremAcc2`, `PremLegal`, `PremTPLM`, `PremTPLV`, `PremServ` and `PremTheft`) to reduce skewness and kurtosis, and then standardize them to apply PCA. The resulting descriptive statistics are reported in Figure 2.5. We can easily observe the numerous high correlations among most of the premium fields. For instance, the linear correlation between `PremLegal` and `PremTPLM` is even greater than 90%, and that between `PremTPLM` and `PremTPLV` is almost 80%. Nonetheless, PCA will return independent variables that would be much more suitable to be used in any regression method. At the same time, such high correlations



	PremWindscreen	PremDamAll	PremFire	PremAcc1	PremAcc2	PremLegal	PremTPLM	PremTPLV	PremServ	PremTheft
mean	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
st. deviation	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
skewness	-1,26	0,10	0,03	0,51	0,87	0,04	0,31	-0,56	-4,79	-0,64
kurtosis	3,61	1,09	1,84	1,27	1,77	3,04	2,91	4,97	242,02	1,74

	PremWindscreen	PremDamAll	PremFire	PremAcc1	PremAcc2	PremLegal	PremTPLM	PremTPLV	PremServ	PremTheft
PremWindscreen	100,0%									
PremDamAll	41,1%	100,0%								
PremFire	66,1%	59,1%	100,0%							
PremAcc1	3,9%	0,3%	4,0%	100,0%						
PremAcc2	0,1%	1,4%	3,0%	-51,0%	100,0%					
PremLegal	31,3%	14,2%	45,5%	5,5%	6,8%	100,0%				
PremTPLM	22,5%	2,2%	33,7%	5,5%	6,3%	90,6%	100,0%			
PremTPLV	32,9%	10,1%	39,3%	4,6%	5,3%	74,0%	79,7%	100,0%		
PremServ	6,2%	0,0%	7,8%	0,6%	1,4%	26,6%	25,4%	21,4%	100,0%	
PremTheft	63,7%	63,9%	93,4%	3,2%	1,9%	27,7%	16,0%	26,5%	2,1%	100,0%

Fig. 2.5: Descriptive statistics of churn rate's explanatory variables

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
st. deviation	1,94	1,42	1,23	0,94	0,75	0,69	0,61	0,52	0,28	0,21
variance %	37,7%	20,2%	15,1%	8,9%	5,6%	4,8%	3,7%	2,8%	0,8%	0,5%
cumulative %	37,7%	57,9%	73,0%	81,9%	87,5%	92,3%	96,0%	98,8%	99,5%	100,0%

Fig. 2.6: Importance of principal components

Stepwise Selection

Step	AIC	Intercept	DriveAge	DrivGender	MaritalStatus	BonusMalus	LicenceNb	PayFreq	JobCode	VehAge	VehClass	VehPower	VehGas	VehUsage	Garage	Area	Channel	PremPC1	PremPC2	PremPC3	PremPC4	OwnedVehiclesAssured
1	10237,44	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2	10227,91	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x
3	10221,76	x	x	x		x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x
4	10215,93	x	x	x		x		x		x	x	x	x	x	x	x	x	x	x	x	x	x
5	10211,21	x	x	x		x		x		x		x	x	x	x	x	x	x	x	x	x	x
6	10207,45	x	x	x		x		x		x		x	x	x	x	x		x	x	x	x	x
7	10204,05	x	x	x		x		x		x		x	x	x		x		x	x	x	x	x
8	10202,13	x	x	x		x				x		x	x	x		x		x	x	x	x	x
9	10200,25	x	x	x		x				x		x	x			x		x	x	x	x	x
10	10198,38	x	x	x		x				x			x			x		x	x	x	x	x
11	**10198,07**	**x**	**x**	**x**		**x**				**x**			**x**			**x**		**x**	**x**		**x**	**x**

Logistic Regression

	Intercept	DriveAge	DrivGender M	BonusMalus	VehAge	VehGasReg ul	AreaA2	AreaA3	AreaA4	AreaA5	AreaA6	AreaA7	AreaA8	AreaA9	PremPC1	PremPC2	PremPC4	OwnedVehiclesAssured2	OwnedVehiclesAssured3
Coefficient	-2,02	0,01	-0,09	0,01	-0,02	-0,16	-1,13	-1,16	-1,03	-1,02	-1,07	-0,98	-0,86	-0,78	0,06	0,06	-0,08	0,43	0,82
Standard Error	0,52	0,00	0,06	0,00	0,01	0,06	0,47	0,47	0,47	0,46	0,52	0,47	0,48	0,47	0,02	0,03	0,03	0,09	0,30
t-Statistic	-3,90	2,67	-1,47	2,89	-1,96	-2,72	-2,39	-2,49	-2,17	-2,20	-2,05	-2,11	-1,79	-1,65	3,00	2,10	-2,97	4,98	2,75
P-Value	0,00	0,01	0,14	0,00	0,05	0,01	0,02	0,01	0,03	0,03	0,04	0,04	0,07	0,10	0,00	0,04	0,00	0,00	0,01
Conf. Interval Lower	-3,12	0,00	-0,20	0,00	-0,03	-0,28	-1,99	-2,02	-1,89	-1,87	-2,06	-1,84	-1,74	-1,64	0,02	0,00	-0,13	0,26	0,19
Conf. Interval Upper	-1,05	0,01	0,03	0,01	0,00	-0,05	-0,11	-0,15	-0,01	-0,02	0,02	0,03	0,17	0,24	0,10	0,11	-0,03	0,59	1,37

Fig. 2.7: Regression summary after stepwise selection

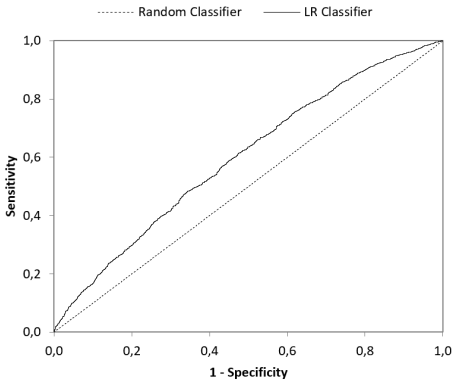


Fig. 2.8: Claim rate ROC

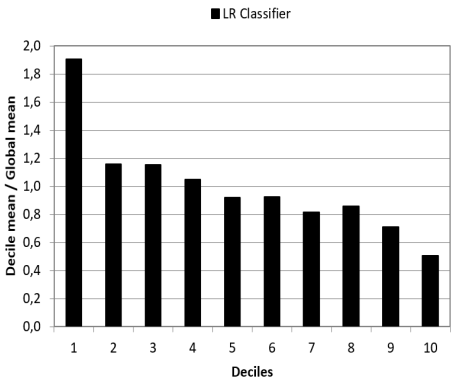


Fig. 2.9: Claim rate deciles

suggest that some principal components could be easily ignored because of the scarce information they represent.
The results of the PCA performed on these premium variables are reported in Figure 2.6, where the principal components are sorted by explained variance. Half of them - from PC6 to PC10 - explain less than 5% information, that is, a small portion. By contrast, the first four principal components - from PC1 to PC4 - explain more than 80% information, which can be considered as a sufficient amount. Further, the first six components explain more than 90% information. In any case, the choice of the number of components is free, and depends on our priorities about parsimony and accuracy. For our analysis, we prefer the former, so we will keep four principal components, defining four new fields: `PremPC1`, `PremPC2`, `PremPC3` and `PremPC4`.
The second part of data preparation relates to the merge of the claim dataset `fremotor1sev` into the main dataset `fremotor1prem`, in order to define the new feature `Claim`, a binary variable equal to 1 if the record reports a claim. More specifically, we need to assign the claims of `fremotor1sev` to the records of `fremotor1prem` by using the field `IDpol`. This is straightforward for unique `IDpol`, but we have already explained in Subsection 2.4.1 that `fremotor1prem` includes duplicates that can correspond to different policyholders and/or vehicles reported under the same `IDpol`. That is the case for a relevant number of records - around 18% of the entire dataset - so we want to keep them, but need a way to assign claims to them, given that there are no duplicates in `fremotor1sev`. To do that, we will fit a logistic regression on the unique records to estimate the claim probability of the duplicated records. After that, we will assign the claims to the duplicated records with the highest estimated claim probability.
Notice that we only assign aggregated claims because `fremotor1sev` does not report any information about frequencies and single claim amounts. Nonetheless, that would have been more relevant for loss estimation or claim reserving, but our analysis is focused on renewal prediction, so we prefer to

keep the minimal information about whether the policyholder made a claim during the year, regardless of any frequency or amount involved in the process. The results of the stepwise logistic regression on the claim rate are reported in Figure 2.7, while the predictive performance is illustrated by the ROC curve and the decile chart in Figure 2.8 (AUC 60%) and 2.9 respectively.
This application of logistic regression should make clear that the distinction between unsupervised and supervised learning is not always as strict as it may seem. Many supervised learning tools are often used for data manipulation (for instance, missing value handling, when they are too frequent to be simply deleted) and dimension reduction (for instance, variable selection using stepwise regression or any other supervised learning tool).

2.4.3 Market basket analysis

As previously described in Subsection 2.3.1, association rules were originally introduced for market basket analysis, which still remains one of their main fields of application. Of course, we could have done more accurate analyses by using more complex tools. However, the fundamental simplicity of association rules in getting and interpreting results make it a good candidate for any market basket analysis.
First and foremost, we need to categorize any numerical variable of our dataset as association rules may be applied to categorical data only. More specifically, we define new categorical fields - `DrivAge_fac`, `BonusMalus_fac`, `VehPower_fac` and `VehAge_fac` - to categorize the related numerical variables (see Figure 2.10).
After that, we need to define binary variables to indicate which record bought each of the guarantees. For that, we can leverage the premium variables: if the premium of one guarantee is nonzero, the related binary variable should be set to 1. Such new variables will be `TPL`, `Legal`, `Serv`, `Fire`, `Theft`, `Windscreen`, `DamAll`, `Acc1` and `Acc2`. Notice that `TPL`, `Legal` and `Serv` are identically 1 in the dataset (they are probably mandatory guarantees), so we should exclude them.
As explained in Subsection 2.3.1, each association rule is determined by an antecedent and a consequent, that is, logical expressions defined by any possible combination of categories in the dataset. To keep it simple, we will avoid complicated expressions by limiting them to one single category for each association rule. Moreover, we will separate the six different guarantees to make it clearer. The results are reported in Figure 2.12-2.17, that is, the twenty most relevant (i.e., highest lift) rules for each guarantee. To avoid insignificant rules, we set a minimum support level to 100 records and a minimum confidence level to 30%. These settings and constraints lead to 435 rules, which are often represented by a scatter plot varying by support, confidence and lift just like in Figure 2.11.

DrivAge_fac

Category	(0,30]	(30,40]	(40,50]	(50,60]	(60,70]	(70,100]
Count	5181	7267	5169	2272	1039	354

BonusMalus_fac

Category	(0,50]	(50,60]	(60,70]	(70,80]	(80,90]	(90,100]	(100,150]
Count	8129	4134	2681	3226	1968	834	310

VehPower_fac

Category	(0,3]	(3,6]	(6,9]	(9,12]	(12,15]
Count	10563	4805	330	286	5298

VehAge_fac

Category	[0,5]	(5,10]	(10,15]	(15,20]	(20,100]
Count	7224	8604	4294	890	270

Fig. 2.10: Variable categorizations for association rules

First of all, notice that the rules for `Fire`, `Theft`, `Windscreen` and `DamAll` (see Figure 2.12-2.15) are quite similar. That is not surprising as they share the same nature. All of them are primarily activated by policyholders to cover their brand new vehicles ("VehAge_fac=[0,5]") or expensive vehicles ("VehClass=Expensive", but also "VehClass=Medium high", "VehClass=Medium" and "VehClass=Medium low").
Other rules are less intuitive. For instance, old policyholders are more likely to buy those types of guarantees. The second highest confidence and lift for `DamAll` relates to over-70 policyholders ("DrivAge_fac=(70,100]"). Actually, we can observe that the antecedents "DrivAge_fac=(50,60]", "DrivAge_fac=(60,70]" and "DrivAge_fac=(70,100]" always appear among the twenty most significant rules for those guarantees. We can conclude that aged drivers tend to be more risk adverse than younger drivers.
The same trend is somewhat explained by the bonus/malus too. Indeed, we can observe the presence of the antecedents "BonusMalus_fac=(0,50]" and "BonusMalus_fac=(50,60]" in each guarantee. Those categories encompass the most expert drivers having the lowest bonus/malus levels: in other words, the less risky customers in portfolio. On the other hand, the fact that the riskiest policyholders are the less prone to insure their own vehicles should be a source of concern for the company, which should try to raise awareness among them.
Less surprising is the presence of "Channel=B". It seems that such a channel is working significantly better than others in selling those guarantees. In this case, the company may try to improve the operations of underperforming channels.
Further, let's focus on `Acc1` (see Figure 2.16) and `Acc2` (see Figure 2.17), which show quite different rules as compared to the aforementioned guar-

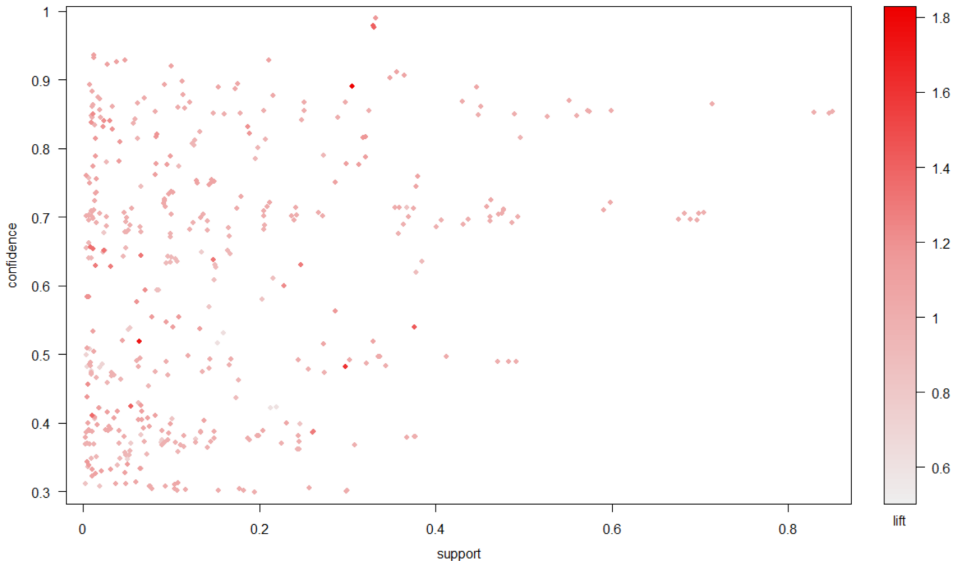


Fig. 2.11: Scatter plot of the 435 association rules

antees. First of all, they substitute each other, because the antecedents "Acc2=N" and "Acc1=N" get the highest lift and the second highest lift respectively.
The location - `Region` and `Area` - seems to represent a major feature to explain the purchase of these guarantees. Indeed, "Region=Paris Area" and "Area=8" get the fourth and fifth highest lifts and confidences for `Acc1`, while "Region=South West" and "Area=7"get the first and third highest lifts and confidences for `Acc2`.
Something interesting to notice is the opposite impact of the bonus/malus as compared to the other guarantees. While the latter had highest lifts by less risky drivers, `Acc1` and `Acc2` show highest lifts by some of the riskiest ones. For instance, the antecedent "BonusMalus_fac=(100,150])" gets the highest lift and confidence for `Acc2` among the bonus/malus categories. However, we should highlight that those lift values are relatively low - 1,15 or lower - so the materiality of the related rules is limited.
A last remark may be done about the fifth and sixth most relevant rules for `Acc2`, that is, "MaritalStatus=Unknown" and "JobCode=Unknown". It may be the case that those 5499 records refer to juridical entities (i.e., firms) rather than physical customers. In such a case, `Acc2` would be the most preferred product to firms for some reasons. It may be a strategic choice of the insurance company, but it may also be a natural consequence of the product design. In the latter case, the insurer should leverage that by offering the same guarantee to other firms.
This subsection was worth to describe a simple way to detect marketing patterns using association rules. Just like renewal prediction itself, association

id	*antecedent*		*consequent*	*support*	*confidence*	*lift*	*count*
1	VehAge_fac=[0,5]	=>	Fire=Y	0,33	0,98	1,40	7047
2	VehClass=Expensive	=>	Fire=Y	0,01	0,85	1,22	197
3	VehClass=Medium	=>	Fire=Y	0,03	0,84	1,20	677
4	VehClass=Medium high	=>	Fire=Y	0,02	0,84	1,20	478
5	VehPower_fac=(3,6]	=>	Fire=Y	0,19	0,83	1,19	3988
6	VehClass=Medium low	=>	Fire=Y	0,09	0,82	1,18	1843
7	VehGas=Diesel	=>	Fire=Y	0,32	0,82	1,17	6814
8	VehPower_fac=(6,9]	=>	Fire=Y	0,01	0,79	1,13	260
9	DrivAge_fac=(60,70]	=>	Fire=Y	0,04	0,78	1,12	812
10	Channel=B	=>	Fire=Y	0,10	0,78	1,12	2037
11	VehAge_fac=(5,10]	=>	Fire=Y	0,31	0,78	1,12	6687
12	DrivAge_fac=(50,60]	=>	Fire=Y	0,08	0,76	1,10	1733
13	VehUsage=Professional run	=>	Fire=Y	0,01	0,76	1,09	131
14	DrivAge_fac=(70,100]	=>	Fire=Y	0,01	0,76	1,09	268
15	BonusMalus_fac=(0,50]	=>	Fire=Y	0,29	0,75	1,08	6123
16	VehClass=Cheap	=>	Fire=Y	0,13	0,75	1,08	2752
17	BonusMalus_fac=(50,60]	=>	Fire=Y	0,15	0,75	1,07	3087
18	Region=Headquarters	=>	Fire=Y	0,15	0,75	1,07	3087
19	Garage=Closed zbox	=>	Fire=Y	0,37	0,74	1,07	7968
20	VehUsage=Professional run	=>	Fire=Y	0,02	0,74	1,06	343

Fig. 2.12: Association rules for `Fire`

id	*antecedent*		*consequent*	*support*	*confidence*	*lift*	*count*
1	VehAge_fac=[0,5]	=>	Theft=Y	0,33	0,98	1,39	7059
2	VehClass=Expensive	=>	Theft=Y	0,01	0,84	1,19	194
3	VehClass=Medium high	=>	Theft=Y	0,02	0,83	1,18	475
4	VehClass=Medium	=>	Theft=Y	0,03	0,83	1,17	668
5	VehPower_fac=(3,6]	=>	Theft=Y	0,19	0,82	1,17	3960
6	DrivAge_fac=(70,100]	=>	Theft=Y	0,01	0,82	1,16	289
7	VehClass=Medium low	=>	Theft=Y	0,09	0,82	1,16	1835
8	VehGas=Diesel	=>	Theft=Y	0,32	0,81	1,16	6798
9	DrivAge_fac=(60,70]	=>	Theft=Y	0,04	0,81	1,15	843
10	VehAge_fac=(5,10]	=>	Theft=Y	0,32	0,79	1,12	6786
11	Channel=B	=>	Theft=Y	0,10	0,79	1,12	2062
12	BonusMalus_fac=(0,50]	=>	Theft=Y	0,30	0,78	1,10	6311
13	VehPower_fac=(6,9]	=>	Theft=Y	0,01	0,78	1,10	256
14	DrivAge_fac=(50,60]	=>	Theft=Y	0,08	0,78	1,10	1761
15	Garage=Closed zbox	=>	Theft=Y	0,38	0,76	1,08	8129
16	Region=Headquarters	=>	Theft=Y	0,15	0,75	1,07	3118
17	BonusMalus_fac=(50,60]	=>	Theft=Y	0,15	0,75	1,07	3106
18	VehUsage=Professional run	=>	Theft=Y	0,01	0,75	1,06	129
19	VehClass=Cheap	=>	Theft=Y	0,13	0,75	1,06	2735
20	Area=A2	=>	Theft=Y	0,10	0,74	1,05	2223

Fig. 2.13: Association rules for `Theft`

id	antecedent		consequent	support	confidence	lift	count
1	VehAge_fac=[0,5]	=>	Windscreen=Y	0,34	0,99	1,16	7131
2	VehClass=Expensive	=>	Windscreen=Y	0,01	0,94	1,10	217
3	DrivAge_fac=(70,100]	=>	Windscreen=Y	0,02	0,93	1,09	330
4	VehPower_fac=(3,6]	=>	Windscreen=Y	0,21	0,93	1,09	4465
5	VehClass=Medium	=>	Windscreen=Y	0,04	0,93	1,09	749
6	DrivAge_fac=(60,70]	=>	Windscreen=Y	0,05	0,93	1,08	962
7	VehClass=Medium high	=>	Windscreen=Y	0,02	0,92	1,08	527
8	VehClass=Medium low	=>	Windscreen=Y	0,10	0,92	1,08	2069
9	VehGas=Diesel	=>	Windscreen=Y	0,36	0,91	1,07	7597
10	BonusMalus_fac=(0,50]	=>	Windscreen=Y	0,35	0,90	1,06	7353
11	VehAge_fac=(5,10]	=>	Windscreen=Y	0,37	0,90	1,06	7780
12	Channel=B	=>	Windscreen=Y	0,11	0,90	1,05	2347
13	VehUsage=Professional run	=>	Windscreen=Y	0,01	0,90	1,05	154
14	DrivAge_fac=(50,60]	=>	Windscreen=Y	0,10	0,89	1,05	2033
15	Region=Headquarters	=>	Windscreen=Y	0,17	0,89	1,05	3693
16	VehClass=Cheap	=>	Windscreen=Y	0,15	0,89	1,04	3255
17	Garage=Closed zbox	=>	Windscreen=Y	0,45	0,89	1,04	9529
18	BonusMalus_fac=(50,60]	=>	Windscreen=Y	0,17	0,89	1,04	3661
19	VehPower_fac=(6,9]	=>	Windscreen=Y	0,01	0,88	1,04	292
20	Region=South West	=>	Windscreen=Y	0,11	0,88	1,03	2400

Fig. 2.14: Association rules for `Windscreen`

id	antecedent		consequent	support	confidence	lift	count
1	VehAge_fac=[0,5]	=>	DamAll=Y	0,30	0,89	1,82	6437
2	DrivAge_fac=(70,100]	=>	DamAll=Y	0,01	0,66	1,34	232
3	VehClass=Expensive	=>	DamAll=Y	0,01	0,66	1,34	152
4	VehClass=Medium	=>	DamAll=Y	0,02	0,65	1,33	526
5	VehClass=Medium low	=>	DamAll=Y	0,07	0,64	1,31	1446
6	VehPower_fac=(3,6]	=>	DamAll=Y	0,14	0,64	1,31	3074
7	VehGas=Diesel	=>	DamAll=Y	0,25	0,63	1,28	5242
8	VehClass=Medium high	=>	DamAll=Y	0,02	0,63	1,28	358
9	DrivAge_fac=(60,70]	=>	DamAll=Y	0,03	0,63	1,28	651
10	BonusMalus_fac=(0,50]	=>	DamAll=Y	0,23	0,60	1,22	4849
11	Channel=B	=>	DamAll=Y	0,07	0,59	1,21	1546
12	VehPower_fac=(6,9]	=>	DamAll=Y	0,01	0,58	1,20	193
13	VehUsage=Professional run	=>	DamAll=Y	0,00	0,58	1,19	100
14	DrivAge_fac=(50,60]	=>	DamAll=Y	0,06	0,57	1,18	1306
15	Garage=Closed zbox	=>	DamAll=Y	0,28	0,56	1,15	6043
16	Area=A2	=>	DamAll=Y	0,08	0,56	1,14	1677
17	Region=Headquarters	=>	DamAll=Y	0,11	0,56	1,14	2302
18	VehClass=Cheap	=>	DamAll=Y	0,09	0,55	1,12	1994
19	BonusMalus_fac=(50,60]	=>	DamAll=Y	0,10	0,54	1,10	2233
20	DrivAge_fac=(40,50]	=>	DamAll=Y	0,13	0,54	1,10	2787

Fig. 2.15: Association rules for `DamAll`

id	antecedent		consequent	support	confidence	lift	count
1	Acc2=N	=>	Acc1=Y	0,38	0,54	1,43	8048
2	OwnedVehiclesAssured=3	=>	Acc1=Y	0,01	0,46	1,21	119
3	JobCode=Other	=>	Acc1=Y	0,01	0,44	1,16	155
4	Region=Paris area	=>	Acc1=Y	0,06	0,43	1,13	1359
5	Area=A8	=>	Acc1=Y	0,02	0,42	1,11	338
6	JobCode=Public employee	=>	Acc1=Y	0,04	0,42	1,10	875
7	OwnedVehiclesAssured=2	=>	Acc1=Y	0,07	0,42	1,10	1396
8	MaritalStatus=Married	=>	Acc1=Y	0,03	0,42	1,10	604
9	MaritalStatus=Cohabiting	=>	Acc1=Y	0,08	0,41	1,09	1808
10	JobCode=Private employee	=>	Acc1=Y	0,07	0,41	1,08	1523
11	MaritalStatus=Single	=>	Acc1=Y	0,02	0,41	1,07	342
12	BonusMalus_fac=(80,90]	=>	Acc1=Y	0,04	0,41	1,07	799
13	BonusMalus_fac=(70,80]	=>	Acc1=Y	0,06	0,40	1,07	1306
14	VehClass=Cheap	=>	Acc1=Y	0,07	0,40	1,07	1473
15	DrivAge_fac=(0,30]	=>	Acc1=Y	0,10	0,40	1,06	2078
16	VehAge_fac=[0,5]	=>	Acc1=Y	0,14	0,40	1,06	2892
17	JobCode=Retiree	=>	Acc1=Y	0,01	0,40	1,05	169
18	VehClass=Medium	=>	Acc1=Y	0,02	0,40	1,05	322
19	Channel=A	=>	Acc1=Y	0,23	0,40	1,05	4913
20	Garage=Closed collective parking	=>	Acc1=Y	0,08	0,40	1,05	1618

Fig. 2.16: Association rules for `Acc1`

id	antecedent		consequent	support	confidence	lift	count
1	Region=South West	=>	Acc2=Y	0,07	0,52	1,72	1413
2	Acc1=N	=>	Acc2=Y	0,30	0,48	1,61	6397
3	Area=A7	=>	Acc2=Y	0,06	0,42	1,41	1189
4	VehUsage=Professional	=>	Acc2=Y	0,01	0,41	1,36	190
5	MaritalStatus=Unknown	=>	Acc2=Y	0,26	0,38	1,28	5499
6	JobCode=Unknown	=>	Acc2=Y	0,26	0,38	1,28	5499
7	BonusMalus_fac=(100,150]	=>	Acc2=Y	0,01	0,35	1,15	107
8	Garage=Street	=>	Acc2=Y	0,05	0,34	1,13	1012
9	LicenceNb=4	=>	Acc2=Y	0,01	0,34	1,13	164
10	Garage=Closed collective parking	=>	Acc2=Y	0,06	0,34	1,12	1368
11	BonusMalus_fac=(80,90]	=>	Acc2=Y	0,03	0,33	1,11	654
12	VehClass=Medium high	=>	Acc2=Y	0,01	0,33	1,10	189
13	Area=A9	=>	Acc2=Y	0,02	0,33	1,10	462
14	Region=Paris area	=>	Acc2=Y	0,05	0,33	1,09	1040
15	BonusMalus_fac=(90,100]	=>	Acc2=Y	0,01	0,33	1,09	272
16	Area=A8	=>	Acc2=Y	0,01	0,32	1,07	259
17	VehAge_fac=[0,5]	=>	Acc2=Y	0,11	0,31	1,04	2265
18	BonusMalus_fac=(50,60]	=>	Acc2=Y	0,06	0,31	1,04	1294
19	BonusMalus_fac=(60,70]	=>	Acc2=Y	0,04	0,31	1,03	832
20	Windscreen=N	=>	Acc2=Y	0,05	0,31	1,03	966

Fig. 2.17: Association rules for `Acc2`

rules represent a common tool to optimize selling and advisory strategies in the most appropriate way.

2.4.4 Churn rate estimation

So far we only used the information in `fremotor1prem` as at year 2003, but we need its records for year 2004 now. Indeed, we define the new binary variable `Churn` to indicate whether one record that are present in 2003 are no longer present in 2004. In this case, `Churn` is set to 1, otherwise 0. The *a priori* churn rate is around 11,3%, which represents a reasonable proportion if compared to the industry average norm in France: retention ratio 82%, that is, churn rate 18% (see Figure 2.1).
As the target variable is binary, this is another problem that may be easily solved by the usual logistic regression. The results are reported in Figure 2.18, while ROC curve and decile chart are shown in Figure 2.19 and 2.20 respectively. The stepwise selects the following churn predictors:

- `DrivAge`
- `DrivGender`
- `BonusMalus`
- `JobCode`
- `VehAge`
- `Garage`
- `Channel`
- `PremPC1`, `PremPC2`, `PremPC3`, `PremPC4`
- `OwnedVehiclesAssured`
- `FamilyMembersAssured`
- `Claim`

with an AUC around 66,4%.
That was what any actuary could easily do to handle the estimation of the churn rate. Nonetheless, let's try to see how clustering may enhance the analysis and improve the results.
First, we need to binarize all the categorical variables, that is, `DrivGender`, `MaritalStatus`, `LicenceNb`, `VehNb`, `PayFreq`, `JobCode`, `VehClass`, `VehGas`, `VehUsage`, `VehGarage`, `Area`, `Region` and `Channel`. Then, the dataset can be partitioned through hierarchical clustering as described in Subsection 2.3.3. Hierarchical clustering partitions the data points for any possible number of

Stepwise Selection

Step	AIC	Intercept	DriveAge	DrivGender	MaritalStatus	BonusMalus	LicenceNb	PayFreq	JobCode	VehAge	VehClass	VehPower	VehGas	VehUsage	Garage	Area	Channel	PremPC1	PremPC2	PremPC3	PremPC4	OwnedVehiclesAssured	FamilyMembersAssured	Claim
1	14362.44	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2	14359,05	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x
3	14355,70	x	x	x		x	x	x	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x
4	14352,51	x	x	x		x	x		x	x	x	x	x	x	x		x	x	x	x	x	x	x	x
5	14350,75	x	x	x		x	x		x	x	x		x	x	x		x	x	x	x	x	x	x	x
6	14349,52	x	x	x		x			x	x	x		x	x	x		x	x	x	x	x	x	x	x
7	14347,44	x	x	x		x			x	x			x	x	x		x	x	x	x	x	x	x	x
8	14346,16	x	x	x		x			x	x				x	x		x	x	x	x	x	x	x	x
9	**14345,26**	**x**	**x**	**x**		**x**			**x**	**x**					**x**		**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**

Logistic Regression

	Intercept	DrivAge	DrivGenderM	BonusMalus	JobCodeFarmer	JobCodeOther	JobCodePrivateemployee	JobCodePublicemployee	JobCodeRetiree	JobCodeUnknown	VehAge	GarageClosed zbox	GarageOpened collective parking	GarageStreet	ChannelB	ChannelL	PremPC1	PremPC2	PremPC3	PremPC4	OwnedVehiclesAssured2	OwnedVehiclesAssured3	FamilyMembersAssured2	Claim
Coefficient	-2,44	-0,01	0,10	0,01	-0,35	-0,14	-0,37	-0,28	-0,53	-0,45	0,01	-0,08	-0,01	0,13	0,68	0,01	-0,09	0,05	0,04	-0,07	0,75	1,12	1,25	-1,24
Standard Error	0,26	0,00	0,05	0,00	0,35	0,23	0,18	0,18	0,24	0,17	0,01	0,06	0,07	0,07	0,07	0,05	0,02	0,02	0,02	0,02	0,05	0,15	0,16	0,14
t-Statistic	-9,29	-2,64	2,09	4,66	-1,00	-0,58	-2,11	-1,53	-2,20	-2,59	2,28	-1,35	-0,13	1,71	9,86	0,14	-5,62	2,22	2,25	-2,70	14,36	7,50	7,95	-8,75
P-Value	0,00	0,01	0,04	0,00	0,32	0,56	0,04	0,13	0,03	0,01	0,02	0,18	0,90	0,00	0,00	0,89	0,00	0,03	0,02	0,01	0,00	0,00	0,00	0,00
Conf. Interval Lower	-2,96	-0,01	0,01	0,01	-1,09	-0,59	-0,71	-0,63	-1,00	-0,77	0,00	-0,20	-0,15	-0,02	0,54	-0,09	-0,12	0,01	0,01	-0,11	0,65	0,82	0,94	-1,53
Conf. Interval Upper	-1,93	0,00	0,20	0,01	0,31	0,32	-0,02	0,09	-0,06	-0,10	0,02	0,04	0,13	0,27	0,81	0,11	-0,06	0,09	0,08	-0,02	0,85	1,40	1,56	-0,97

Fig. 2.18: Regression summary after stepwise selection

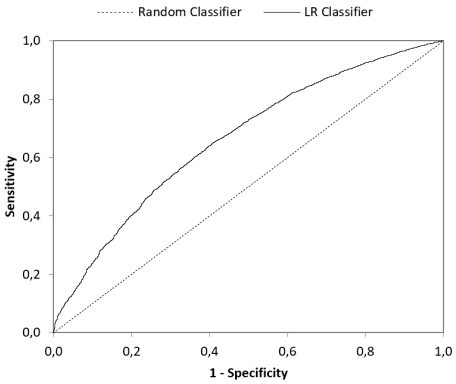


Fig. 2.19: Churn rate ROC

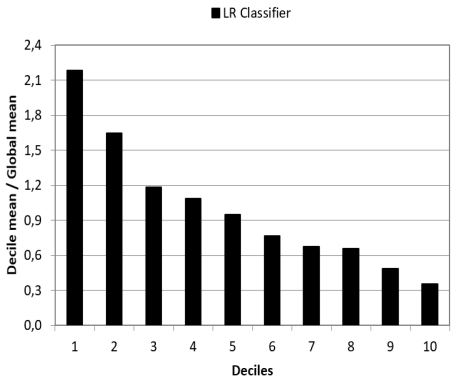


Fig. 2.20: Churn rate deciles

clusters, so we get finer partitions and lower within variance as that number increases.

Looking at the reduction in within variance in Figure 2.21 represents an intuitive way to decide the number of clusters. It is not always an obvious choice, but in this case we observe major reductions until the seven-cluster partition (indicated by the dashed line in Figure 2.21), while they get less material onwards. Figure 2.22 illustrates the dendrogram as well as the seven-cluster choice.

Furthermore, this method does not help only with the choice of the number of clusters for the hierarchical methods, but also for other methods. For instance, as suggested by numerous recent studies (see Arai et al. (2007), Chen et al. (2005) and Liu et al. (2017), among others), we will use the same number of clusters for the K-means method starting from the cluster centroids returned by the hierarchical clustering itself. That tackles the two major drawbacks of K-means clustering. First, it provides us with a solid reason to set $K = 7$. Second, it fixes the problem of the initial conditions at once and deterministically. Actually, we might have run the K-means algorithm several times with random initial conditions in order to produce stable clusters (this is the approach needed to produce the related within variance reduction in Figure 2.23, for which 10 different initial conditions are applied to each value for K, between 1 and 100), but it would have been much heavier from a computational perspective. On the other hand, K-means clustering is so attractive thanks to its low runtime above all. All in all, the suggested approach avoids the repetitive usage of the costly algorithm behind hierarchical clustering: we need to run it only once to get the inputs for the K-means method, and then we can always use the latter as much as we want and as long as we assume that those inputs are still relevant.

To better compare the differences between the two clustering methods in terms of potential in predicting renewals, we run one logistic regression for

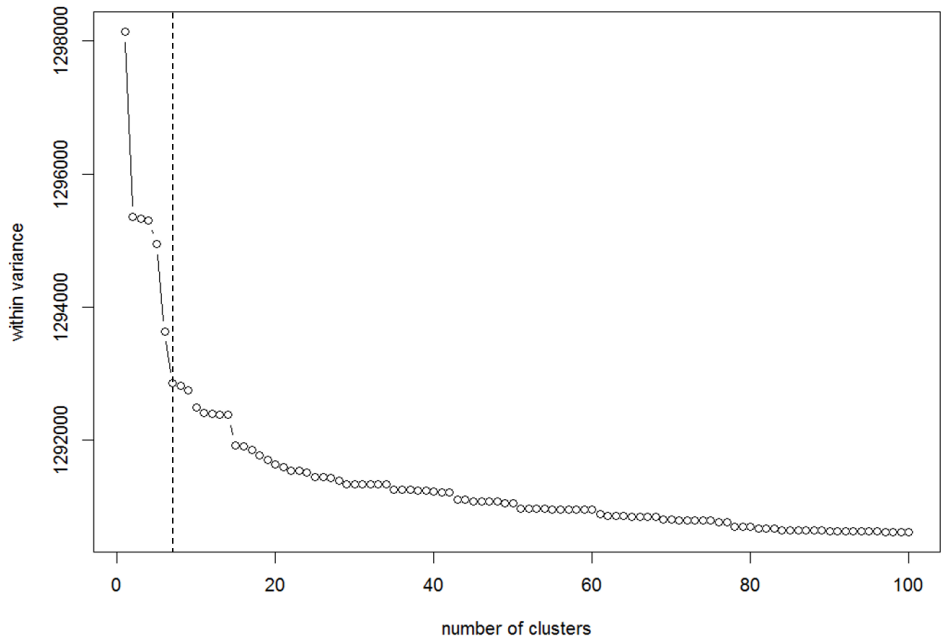


Fig. 2.21: Within variance by number of hierarchical clusters

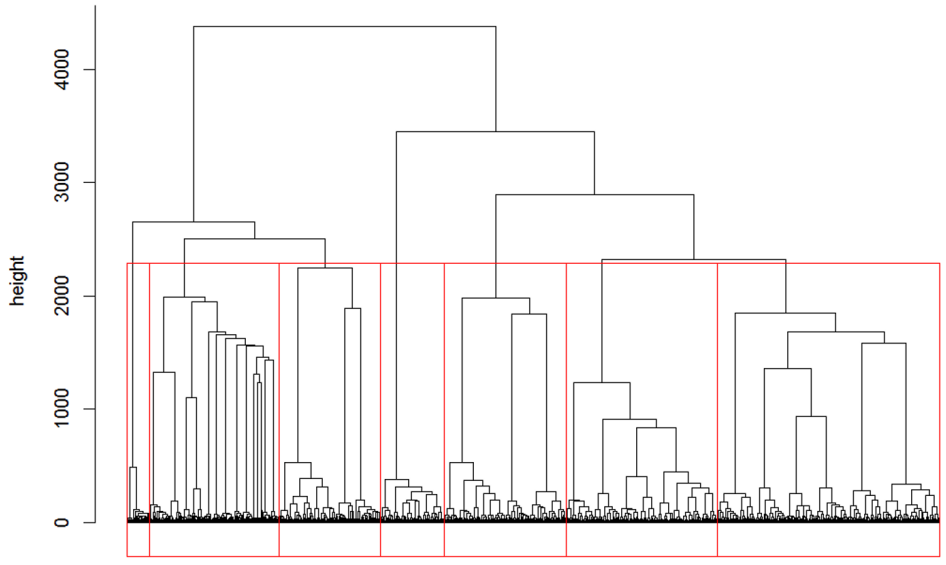


Fig. 2.22: Dendrogram and seven-cluster split

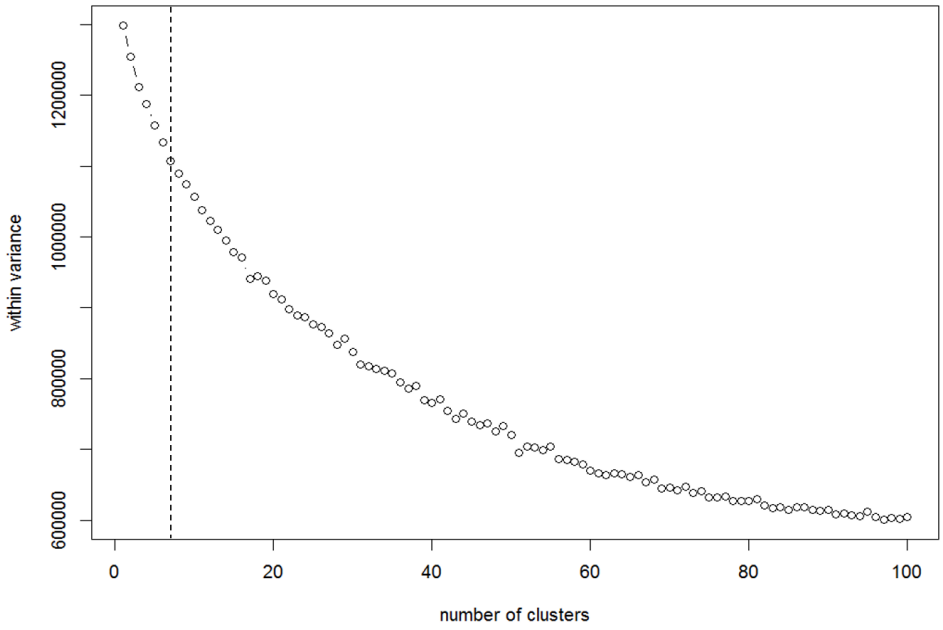


Fig. 2.23: Within variance by number of K-means clusters

each cluster in order to observe the differences with the logistic regression that was previously run on the non-clustered dataset.
Figure 2.24 summarizes the parameter estimates returned by the logistic regressions on the seven hierarchical clusters. The bold estimates are the most significant ones, with a p-value lower than 5%. Some predictors are so correlated with the churn rate that they are kept in most of the clusters. They include `BonusMalus` (within all the clusters but one), `PremPC1` (within all the clusters but two), `OwnedVehiclesAssured` (within all the clusters but one) and `Claim` (within all the clusters). Not surprisingly, those variables are kept in the no-clustering regression as well. Nonetheless, less present variables are still kept by that, for instance, `DrivGender` (in the third cluster only), `JobCode` (in the fifth cluster only), `VehAge` (in the second cluster only), `Garage` (in the fifth cluster only) and `Area` (in the seventh cluster only). More importantly, there are few predictors that do not appear in the no-clustering regression, although they do appear in some clusters. For example, `PayFreq` is quite significant in both the fourth and seventh cluster, but it is not in the no-clustering regression. This occurs to `VehClass`, `VehGas`, `VehUsage` and `Area` as well. To some extent, it could represents a loss of relevant information when we do not account for clusters.
Similar remarks can be done for the seven clusters produced by the K-means algorithm. Figure 2.25 summarizes the parameter estimates returned by the logistic regressions on them. The bold estimates are the most signifi-

Variable	Category	cluster 1	cluster 2	cluster 3	cluster 4	cluster 5	cluster 6	cluster 7	no clustering
Intercept	-	-2,5249	-3,1517	-3,4540	-3,0342	-3,2449	-2,9558	-17,1167	-2,4369
DrivAge	-	-0,0064	-	-	-	-0,0104	-	-	-0,0061
DrivGender	M	-	-	0,2025	-	-	-	-	0,1007
MaritalStatus	Divorced	-	-	-	-	-	-	-	-
MaritalStatus	Married	-	-	-	-	-	-	-	-
MaritalStatus	Single	-	-	-	-	-	-	-	-
MaritalStatus	Widowed	-	-	-	-	-	-	-	-
MaritalStatus	Unknown	-	-	-	-	-	-	-	-
BonusMalus	-	0,0076	0,0096	0,0154	0,0097	0,0200	0,0172	-	0,0096
LicenceNb	2	-	-	-	-	-	-	-	-
LicenceNb	3	-	-	-	-	-	-	-	-
LicenceNb	4	-	-	-	-	-	-	-	-
LicenceNb	5	-	-	-	-	-	-	-	-
PayFreq	Half-yearly	-	-	-	0,3476	-	-	0,4611	-
PayFreq	Monthly	-	-	-	-0,0891	-	-	-14,6481	-
PayFreq	Quarterly	-	-	-	0,3870	-	-	-14,8979	-
JobCode	Farmer	-	-	-	-	-	-	-	-0,3538
JobCode	Other	-	-	-	-	-	-	-	-0,1359
JobCode	Private employee	-	-	-	-	-	-	-	-0,3738
JobCode	Public employee	-	-	-	-	1,6899	-	-	-0,2795
JobCode	Retiree	-	-	-	-	-	-	-	-0,5264
JobCode	Unknown	-	-	-	-	-0,5877	-	-	-0,4465
VehAge	-	-	0,0511	-	-	-	-	-	0,0133
VehClass	Cheaper	-	-	-0,1392	-	-	-	-	-
VehClass	Cheapest	-	-	0,1131	-	-	-	-	-
VehClass	Expensive	-	-	-	-	-	-	-	-
VehClass	Medium	-	-	17,1885	-	-	-	-	-
VehClass	Medium high	-	-	-	-	-	-	-	-
VehClass	Medium low	-	-	0,1346	-	-	-	-	-
VehClass	More expensive	-	-	-	-	-	-	-	-
VehClass	Most expensive	-	-	-12,0003	-	-	-	-	-
VehPower	-	-	-	-	-	-	-	-	-
VehGas	Regular	-	-	-	-	0,2491	-	-	-
VehUsage	Professional	-	-	-	-	-	-	-	-
VehUsage	Professional run	-	-	-	-	-	-0,5577	-	-
Garage	Closed zbox	-	-	-	-	-0,1015	-	-	-0,0810
Garage	Opened collective	-	-	-	-	0,0842	-	-	-0,0097
Garage	Street	-	-	-	-	0,4169	-	-	0,1272
Area	A2	-	-	-	-	-	-	14,4748	-
Area	A3	-	-	-	-	-	-	15,2946	-
Area	A4	-	-	-	-	-	-	15,2671	-
Area	A5	-	-	-	-	-	-	15,1244	-
Area	A6	-	-	-	-	-	-	-	-
Area	A7	-	-	-	-	-	-	15,0418	-
Area	A8	-	-	-	-	-	-	-	-
Area	A9	-	-	-	-	-	-	47,8231	-
Channel	B	-	-	0,6868	0,5885	-9,8224	0,9839	-	0,6799
Channel	L	-	-	-0,0985	-0,1957	0,3363	-0,1283	-	0,0072
PremPC1	-	-0,0958	-	-0,1535	-0,1010	-0,1666	-	-0,0596	-0,0859
PremPC2	-	0,0942	-	-	0,1127	-	-	0,1241	0,0457
PremPC3	-	-	0,1301	-	-	-	-	0,1379	0,0413
PremPC4	-	-0,1098	-	-0,1072	-	-	-	-	-0,0657
OwnedVehiclesAssured	2	0,7334	0,6571	0,8004	0,7565	1,2254	-	0,3580	0,7523
OwnedVehiclesAssured	3	-	1,0202	-	-	-	-	-	1,1175
FamilyMembersAssured	2	-	1,1695	-	-	-	-	-	1,2541
Claim	1	-1,4465	-0,7125	-1,0770	-1,2101	-1,6761	-1,5716	-1,5731	-1,2397

Fig. 2.24: Logistic regression coefficients by hierarchical cluster

Variable	Category	cluster 1	cluster 2	cluster 3	cluster 4	cluster 5	cluster 6	cluster 7	no clustering
Intercept	-	-2,9947	-4,0221	-3,3187	-3,6767	-1,1097	-3,2127	-15,7628	-2,4369
DrivAge	-	-0,0073	-	-	-	-0,0105	-	-	-0,0061
DrivGender	M	-	0,5372	-	0,2724	-	-	-	0,1007
MaritalStatus	Divorced	-	-	-	-	-	-	-	-
MaritalStatus	Married	-	-	-	-	-	-	-	-
MaritalStatus	Single	-	-	-	-	-	-	-	-
MaritalStatus	Widowed	-	-	-	-	-	-	-	-
MaritalStatus	Unknown	-	-	-	-	-	-	-	-
BonusMalus	-	0,0131	-	0,0141	0,0196	0,0168	0,0216	-	0,0096
LicenceNb	2	0,0306	-	-	-	-	-	-	-
LicenceNb	3	-	-	-	-	-	-	-	-
LicenceNb	4	-	-	-	-	-	-	-	-
LicenceNb	5	1,7766	-	-	-	-	-	-	-
PayFreq	Half-yearly	-	-0,0521	-	-	-	-	0,3638	-
PayFreq	Monthly	-	0,2214	-	-	-	-	-13,8041	-
PayFreq	Quarterly	-	0,8125	-	-	-	-	-0,6296	-
JobCode	Farmer	-	-14,2842	-	-	-1,4038	-	-	-0,3538
JobCode	Other	-	0,9946	-	-	-	-	-	-0,1359
JobCode	Private employee	-	0,6006	-	-	-0,6532	-	-	-0,3738
JobCode	Public employee	-	0,7304	-	-	-0,8076	-	-	-0,2795
JobCode	Retiree	-	-13,6777	-	-	-0,7100	-	-	-0,5264
JobCode	Unknown	-	0,8710	-	-	-0,9401	-	-	-0,4465
VehAge	-	0,0226	0,0486	-	-	-	-	-	0,0133
VehClass	Cheaper	-	-0,0840	-0,1545	-	-0,0689	-	-	-
VehClass	Cheapest	-	0,2999	0,0886	-	-0,2621	-	-	-
VehClass	Expensive	-	1,4336	0,9724	-	1,1720	-	-	-
VehClass	Medium	-	-0,2563	0,6320	-	-0,2789	-	-	-
VehClass	Medium high	-	-1,5888	0,5425	-	0,4564	-	-	-
VehClass	Medium low	-	-0,5683	0,2056	-	0,0824	-	-	-
VehClass	More expensive	-	-0,8822	0,6996	-	-0,6977	-	-	-
VehClass	Most expênsive	-	-16,6690	0,5581	-	0,8683	-	-	-
VehPower	-	-	-	-	-	-	-	-	-
VehGas	Regular	-	-	-	-	0,1463	-	-	-
VehUsage	Professional	-	-	-	-	-	-	-	-
VehUsage	Professional run	-	1,6956	-	-	-	-	-	-
Garage	Closed zbox	-	-	-0,0837	-	-0,0372	-	-0,2622	-0,0810
Garage	Opened collective	-	-	0,2768	-	0,0305	-	-0,2463	-0,0097
Garage	Street	-	-	0,2686	-	0,3219	-	0,1256	0,1272
Area	A2	-	-	-	-	-1,2154	-	13,6046	-
Area	A3	-	-	-	-	-1,2760	-	14,1807	-
Area	A4	-	-	-	-	-1,4637	-	14,1522	-
Area	A5	-	-	-	-	-1,2591	-	13,9141	-
Area	A6	-	-	-	-	-1,6779	-	14,1287	-
Area	A7	-	-	-	-	-0,9685	-	13,7657	-
Area	A8	-	-	-	-	-1,5217	-	14,4391	-
Area	A9	-	-	-	-	0,9553	-	13,6955	-
Channel	B	-	-	-	0,7567	-	0,8702	-	0,6799
Channel	L	-	-	-	-0,1789	-	-0,2771	-	0,0072
PremPC1	-	-0,0717	-	-0,1328	-0,1797	-0,1911	-	-	-0,0859
PremPC2	-	-	0,0992	-	-	-	-	0,1490	0,0457
PremPC3	-	-	0,1488	-	-	-	-	0,1506	0,0413
PremPC4	-	-	-	-0,1372	-	-	-	-	-0,0657
OwnedVehiclesAssured	2	0,7482	0,9975	0,6994	0,9668	0,9665	-	-	0,7523
OwnedVehiclesAssured	3	1,0850	0,5244	1,6161	1,0757	1,2900	-	-	1,1175
FamilyMembersAssured	2	-	1,7539	-	1,3442	-	-	1,2388	1,2541
Claim	1	-1,1021	-	-1,1096	-1,5095	-1,4660	-2,0418	-1,5083	-1,2397

Fig. 2.25: Logistic regression coefficients by K-means cluster

	cluster	1	2	3	4	5	6	7	**Overall**
hierarchical clustering	count	5824	3405	3201	2652	3957	568	1675	21282
	AUC	65,1%	67,2%	66,2%	66,4%	71,2%	68,0%	64,5%	**66,9%**
K-means clustering	count	4269	1366	3612	1738	7409	465	2423	21282
	AUC	65,0%	75,2%	65,9%	68,4%	69,3%	67,1%	64,7%	**67,6%**
no clustering	count	21282							21282
	AUC	66,4%							**66,4%**

Fig. 2.26: Performance comparison by AUC

cant ones, with a p-value lower than 5%. Some predictors are so correlated with the churn rate that they are kept in most of the clusters. They include `BonusMalus` (within all the clusters but two), `OwnedVehiclesAssured` (within all the clusters but two) and `Claim` (within all the clusters but one). Not surprisingly, those variables are kept in the no-clustering regression as well. Nonetheless, less present variables are still kept by that, for instance, `DrivAge` (in the first and fifth cluster only), `DrivGender` (in the second and fourth cluster only), `JobCode` (in the second and fifth cluster only), `VehAge` (in the first and second cluster only) and `Area` (in the fifth and seventh cluster only). More importantly, there are few predictors that do not appear in the no-clustering regression, although they do appear in some clusters. For example, `PayFreq` is quite significant in both the second and seventh cluster, but it is not in the no-clustering regression. This occurs to `LicenceNb`, `VehClass`, `VehGas`, `VehUsage` and `Area`, that is, the same variables as in the hierarchical clustering except for `LicenceNb`. It represents one more indication on the loss of information caused by their exclusions in the no-clustering regression.

The simplest way to observe whether - and how much - the no-clustering regression is missing in terms of predictive power is based on the AUC values reported in Figure 2.26. The overall AUCs for the two clustering methods are calculated as averages of the single AUCs weighted by the number of records in each cluster. Apparently, they can explain slightly more information than the no-clustering regression, that is, further 0,5% and 1,2% for hierarchical clustering and K-means clustering respectively. Even if this gain may not necessarily justify the cluster analysis itself, it still shows that single regressions on specific clusters can materially outperform the no-clustering regression. For instance, the K-means performance on the second cluster exceeds 75% AUC, much higher than 66,4%. Quite significant is the performance on the fifth hierarchical cluster, which exceeds 71%. In other words, it helps us to identify clusters of policyholders whose renewal may

be anticipated more accurately.
Before concluding this section, it is worth taking a look at the practical impact of each relevant predictor on the churn rate. More specifically, we will use the best method with the highest AUC, that is, the logistic regression combined with K-means clustering (see Figures 2.26). The results - distributions and churn rate estimations - are illustrated in Figure 2.27-2.38.
The most relevant, policyholder-related predictors include driver gender (`DrivGender`), driver age (`DrivAge`), job type (`JobCode`) and sale channel (`Channel`). Figure 2.27 shows that men lapse significantly more often than women, with a churn rate higher than 13,5% against a churn rate lower than 11%. At the same time, the youngest drivers have a 15% churn rate at least (between 18 and 20 years old it is even higher than 20%!), which should be something to be appropriately managed by the company (see Figure 2.28). However, it is somewhat balanced by older drivers, with a churn rate stabilized below 10%. This is also confirmed by the job impact: retired policyholders show the lowest churn rate - below 10% - among the seven categories of `JobCode` (see Figure 2.29). Moreover, as shown in Figure 2.30, the sale channel seems to have a material impact for the category indicated by "B" (unfortunately, we do not have any specific information about the meaning of the `Channel` categories).
The most relevant, vehicle-related predictors include vehicle age (`VehAge`) and garage type (`Garage`). In Figure 2.31, we can observe a linear shape of the churn rate by vehicle age. In particular, new vehicles tend to relate to low churn rates, approximately lower than 11%, while churn rate seems to increase as they get older. However, we should highlight the huge variability of results at high vehicle ages, which prevent us from giving accurate estimation. A linear shape is also shown in Figure 2.32, that is, the more exposed the parking type, the higher the churn rate. For instance, policyholders that own a closed box have a churn rate around 11%, while policyholders parking on the street have a churn rate around 14%.
As expected, premium amounts are quite relevant in predicting renewal. In order to avoid any impractical analysis on the principal component of the premiums in the dataset, we will analyse the change in churn rate by overall third-party premium (see Figure 2.33) and overall damage premium (see Figure 2.34). The former is clearly another churn driver: it slowly makes the churn rate higher and higher at premiums below 1000, while it lets it explode at greater amounts. The latter does not impact so much, probably because the related amounts are much lower, so less significant for the policyholders: except at very low premium amounts, the churn rate is always around 10%.
The riskiness of the policyholder plays a role as well, although it is less intuitive to figure out. Figure 2.35 clearly shows that the churn rate increases as the bonus/malus category of the policyholder (i.e., the potential riskiness) increases. From a business perspective, this is perfectly fine: for some rea-

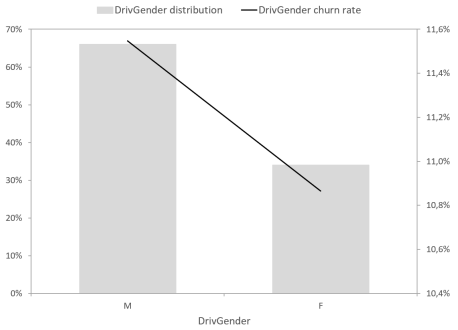


Fig. 2.27: Churn rate by driver gender

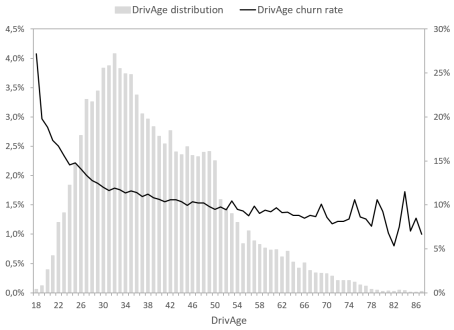


Fig. 2.28: Churn rate by driver age

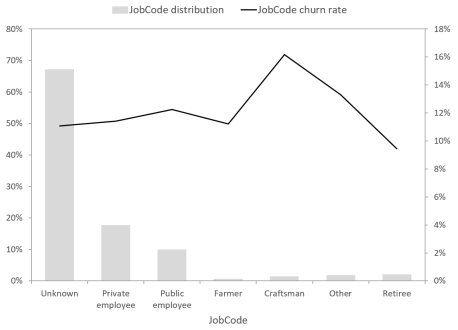


Fig. 2.29: Churn rate by job code

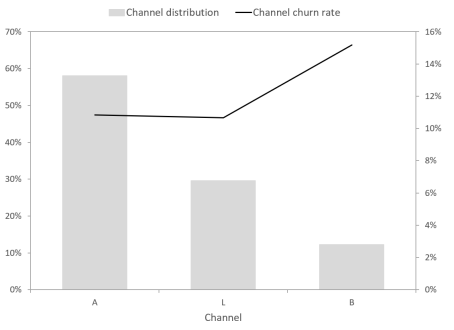


Fig. 2.30: Churn rate by channel

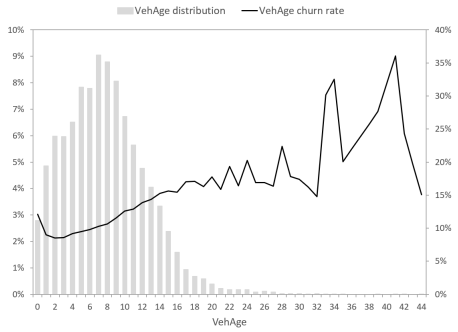


Fig. 2.31: Churn rate by vehicle age

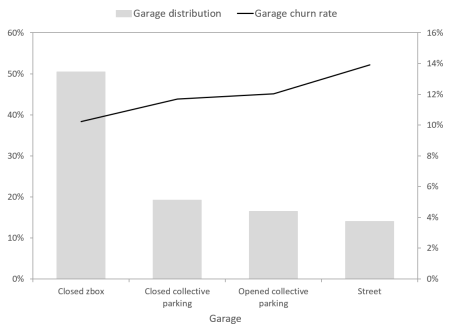


Fig. 2.32: Churn rate by garage

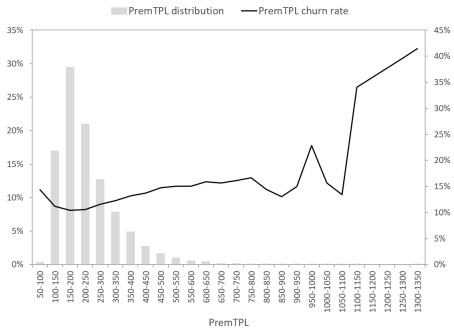


Fig. 2.33: Churn rate by third-party premium

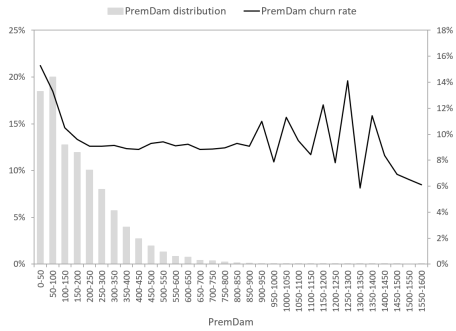


Fig. 2.34: Churn rate by damage premium

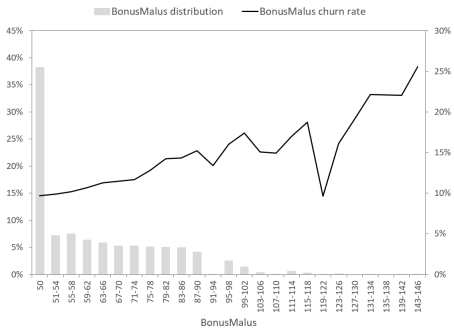


Fig. 2.35: Churn rate by bonus-malus

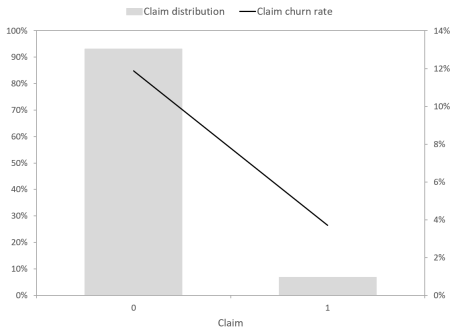


Fig. 2.36: Churn rate by claim occurrence

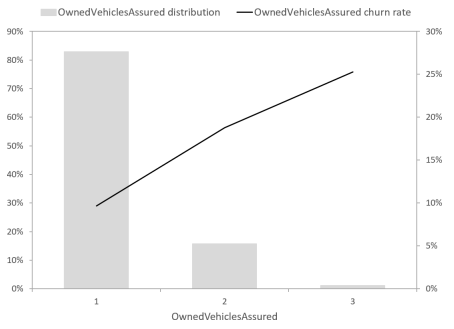


Fig. 2.37: Churn rate by vehicles assured

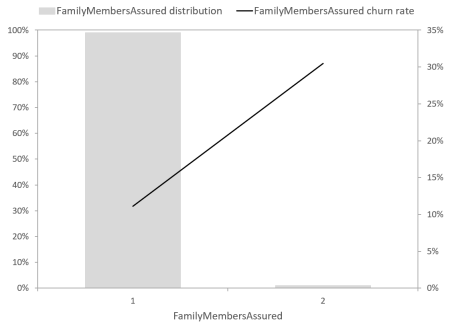


Fig. 2.38: Churn rate by members assured

sons, riskier policyholders tend to lapse more often. However, that seems to be in contrast to what is shown in Figure 2.36, that is, policyholders that caused a claim in 2003 will lapse in 2004 with an estimated likelihood around 4%, which is extremely low. In other words, we should separate the *historical* information provided by `BonusMalus` from the *actual* information provided by `Claim`: even if both of them are two expressions of the same risk, their impact on the churn rate is opposite.
Finally, the number of vehicles assured (see Figure 2.37) and members assured (see Figure 2.38) are relevant in the same direction. In other words, the more the vehicles or members assured, the higher the chance of lapse. This is simply due to the fact that the presence of more available vehicles make all of them less necessary, so it is reasonable not to expect the renewal of one (or even more) of the policies with a higher chance, especially when one vehicle has just replaced another vehicle.

2.5 Limitations, extensions and conclusions

The main purpose of this chapter is the demonstration of how unsupervised learning can be used to support data preparation and/or enhance customer behaviour prediction. According to the CRISP-DM standard for the data mining process (see Section 1.2), the chapter can be formally broken as follows:

C1 *Business understanding*: Section 2.1

C2 *Data understanding*: Subsection 2.4.1

C3 *Data preparation*: Subsection 2.4.2, 2.4.3 and 2.4.4

C4 *Modelling*: Subsection 2.4.3 and 2.4.4

C5 *Evaluation*: Subsection 2.4.3 and 2.4.4

C6 *Deployment*: none.

Unlike the applications of the next two chapters, this one is not actuarial in nature, but it is however necessary to suggest the importance of data by introducing unsupervised learning, and build a bridge to the next topics. Nonetheless, the results produced may still be useful for several actuarial exercises. For instance, premium loadings may be calibrated and customized taking into account the likelihood of renewal for each policyholder in the next few years, in order to avoid unexpected losses. Moreover, we may use the estimated churn rates to evaluate the profitability of a portfolio on a more realistic basis and over a time horizon longer than one year. That could be considered as a sort of life-style evaluation applied to non-life.
In the next chapters, we will introduce some of the most common supervised

learning methods in data science. They will play the same role that logistic regression played in this chapter, that is, tools to predict target variables. As we will also demonstrate, those methods can outperform logistic regression as they tend to adapt to data. Of course, we could have tried to use them to predict churn rate too, but, as explained, supervised learning was not the primary purpose of this chapter, so we will start employing them from the next one.

2.5.1 Key conclusions for actuarial practitioners

Data scientists usually leverage unsupervised learning to simplify and improve subsequent supervised learning. If an actuary intends to apply data science, he/she needs to know that range of techniques. As we explained in this chapter, indeed, unsupervised learning can help in several ways.
While preparing the dataset, one may need to compress data in order to reduce size without losing too much information. There are a lot of data reduction techniques: we introduced the most common ones. To reduce the columns, actuaries can apply PCA as well as its adaptations for nonnumerical variables, just like we did in this chapter for the premium amount fields. Essentially, we could halve them. Instead, to reduce the rows, cluster analysis can help by grouping similar records based on the distance measure we defined. In our analysis, we did not need to use that for dimension reduction as the number of records was fairly manageable, but clustering can be crucial when this is not the case.
Nonetheless, we demonstrated how cluster analysis may improve performance of supervised learning tools. Indeed, supervised learning on single clusters slightly outperformed supervised learning on the entire dataset. Of course, this is not always the case, because it depends on the level of heterogeneity into the dataset. However, it shows that unsupervised learning techniques are not merely used for more complex data handling, but they can even boost supervised learning performance.

Chapter 3

Individual Claim Reserving using Supervised Learning

This chapter will be the first one dealing with supervised learning. More specifically, a number of techniques will be introduced, distinguishing between classification and prediction algorithms. They include naïve Bayes classifiers, nearest neighbours, decision trees and neural networks, which represent the most widely used supervised learning techniques besides GLMs.

Actuarial context

The application relates to a classical problem in non-life actuarial practice, that is, claim reserve estimation. Actuaries mostly deal with it by aggregating claim payments to calculate the overall reserve. This is usually done with run-off triangles, which account for accident year, reporting year and closing delay. This has been proven to return accurate estimations as long as claims are homogeneous and their payment is not correlated with other features. However, this is not always the case, and sometimes individual reserving is necessary for a better reserving process. Generally speaking, individual reserving is performed for each single claim by using all the available information, in contrast to aggregation methods such as run-off triangles.

Chapter overview

Given that the data-driven nature of machine learning can be more easily leveraged by individual reserving, this is the approach we will choose for this chapter. The dataset consists of automobile bodily injury claims from the Australian insurance market, which guarantees a high level of heterogeneity, and thus a potential outperformance of more flexible machine learning methods over GLMs.
The analysis will be split into two separate exercises: the classification among closing delay classes and the prediction of claim payment amount.

Both of them will be tackled by using all the aforementioned machine learning techniques as well as proper GLMs (i.e., multinomial regression and gamma regression respectively) in order to compare their performances. In closing delay estimation, all the methods will approximately return the same accuracy, so it does not justify the usage of alternative techniques. On the contrary, we will demonstrate that the claim payment amount predictions returned by the decision tree are significantly more accurate than those of the GLM.
Even if it represents just one specific case study that cannot be generalized, the chapter will emphasize the additional predictive power that more flexible techniques may offer, that is, better exploitation of data variability.

3.1 Introduction

Actuarial practice in non-life reserving is traditionally based on aggregate claims data structured in triangles. In fact, this has been proven (for instance, in Bonrhuetter et al. (1972) and Mack (1993)) to be an effective approach as long as we face high-probability low-impact claims such as those of motor insurance. Run-off triangles - like that in Figure 3.1 - are fundamentally based on the assumption that the reserve on future claim payments depend on accident year, reporting year and closing delay only. Actually, it is far to be true, unless the claims tend to be extremely homogeneous, and even in that case, triangle-based estimations will neglect relevant information about individual claims. This was necessary when actuaries had to use data in times of strong computational limits. Nowadays, this is no longer a major constraint.
On the other hand, in parallel to triangle-based methods for the total claim reserve, insurance companies employ *case reserving* methods to estimate individual claim reserves, that is, the expected ultimate loss. The first reserve estimate, typically input within a few days of receiving claim notice, is based on the little information known at that time. As the claim matures, this estimated value should be established for the claim based on the information known at any point, net of any partial amount previouly paid to the policyholder. This individual approach allows for a continuous monitoring of the single case, and may be much more useful for low-probability high-impact claims.
Thanks to the current data availability, many studies on claim reserving published in the last ten years promote the usage of individual claim data for aggregated reserving as well as case reserving. Some of the most recent include Taylor et al. (2008), Jessen et al. (2011), Pigeon et al. (2013), Antonio et al. (2014), Martinez-Miranda et al. (2015), Antonio et al. (2016), Avanzi et al. (2016), Badescu et al. (2016) and Hiabu et al. (2016). To some extent, all of them assume a rather fixed structural form for the timing or

reporting year	*closing delay* 0	1	...	$n-1$	n	*tail*
$T-n$	$P_{T-n,0}$	$P_{T-n,1}$	...	$P_{T-n,n-1}$	$P_{T-n,n}$	$\widetilde{P}_{T-n,c}$
$T-(n-1)$	$P_{T-(n-1),0}$	$P_{T-(n-1),1}$	...	$P_{T-(n-1),n-1}$	$\widetilde{P}_{T-(n-1),n}$	$\widetilde{P}_{T-(n-1),c}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
$T-1$	$P_{T-1,0}$	$P_{T-1,1}$	...	$\widetilde{P}_{T-1,n-1}$	$\widetilde{P}_{T-1,n}$	$\widetilde{P}_{T-1,c}$
T	$P_{T,0}$	$\widetilde{P}_{T,1}$	...	$\widetilde{P}_{T,n-1}$	$\widetilde{P}_{T,n}$	$\widetilde{P}_{T,c}$

Fig. 3.1: Run-off triangle of paid amounts, and related reserve estimations (in red)

the amount of the payments, often based on some GLM form. To relax the rigidity due to full parametric models like those, machine learning may represent a further option to explore.
It is worth highlighting that machine learning can be easily and directly applied to case reserving as it relates to individual estimations. Another intuitive application of machine learning in non-life practice involves pricing (see Wüthrich et al. (2018) for a detailed presentation of the topic), which is individual by definition. On the contrary, the application of the same techniques to triangle-based reserving is less obvious, because it is based on aggregated data, and takes into account a limited number of features. In the recent years, however, some researchers tried to enhance such traditional methods by using machine learning (among others, see Wüthrich (2016b), which applies classification trees to estimate number of future payments varying by accident year and reporting delay, and Wüthrich (2017), which applies neural networks to handle heterogeneity in data and improve Chain-Ladder reserving).
On our side, we will try to extend those ideas by estimating the case reserve by using a range of machine learning tools, in order to emphasise their advantages (in terms of accuracy above all) as well as their drawbacks. Even if we will not deal with triangle-based methods, the results will be reported in the traditional reporting-year-per-closing-delay format, based on the two target variables of our application, that is, the closing delay and the ultimate cost.
After a brief introduction to the reserving problem in non-life insurance (see Section 3.2) and the probabilistic model at the base of our analysis (see Section 3.3), we will recall some features of the machine learning tools we intend to use (see Section 3.4). All of them will be applied to publicly available automobile bodily injury claim data, in order to estimate individual case reserves. The analysis is presented in Section 3.5. In line with the

underlying model, the results will be broken down into three steps:

1. closing delay estimation of individual claims through multinomial regression, naïve Bayes, nearest neighbours and classification tree (see Subsection 3.5.2)

2. payment amount estimation of individual claims through generalized regression, regression tree and neural network (see Subsection 3.5.3)

3. case reserve estimation of individual claims through some combinations between tools in 1. and tools in 2. (see Subsection 3.5.4).

Furthermore, we will also address the problem of the separation between small claims and large claims (see Subsection 3.5.5), which is part of the non-life best practice, and may be useful for various reasons.
While no machine learning tool will be able to explain a relevant amount of variance at the closing delay's step, payment amount estimations and claim separation performances will be quite accurate. More specifically, the estimations returned by the decision tree and the neural network will be significantly more accurate than those from a GLM based on the gamma distribution. Of course, this is not to prove that machine learning can always outperform more traditional approaches. Nonetheless, it should provide actuaries with further techniques to better estimate individual reserves when it comes with heterogeneous data.

3.2 Understanding claim timeline

Generally, a claim is triggered by an accident causing a damage covered by the insurance contract. In an ideal world, the related benefit is paid as soon as the accident occurs, but often this is not the case in non-life insurance. In fact, a number of years may pass between the effective occurrence and the final claim payment (or payments). This time gap represents the reason why insurance companies must allocate reserve sufficient to cover any future payments for outstanding loss liabilities.
Assume the premium is paid in t_0 for an insurance protection that is immediately effective for a period T. During that period, an accident occurs at time $t_a < T$, the so-called *accident date*. Ideally, the accident is immediately reported to the company, but for a number of reasons it may happen differently, that is, the accident is reported at any time $t_r \geq t_a$, the so-called *reporting date*. The difference $\Gamma := t_r - t_a$ is the *reporting delay*. If it is small, say days, it does not really represent a problem to the company. However, if the reporting delay extends for years, it generates an unknown outstanding loss liability for the company, which is backed by the so-called *Incurred-But-Not-Yet-Reported reserve* - or IBNYR reserve. Actually, the related claims are not in the company's systems yet, so data-driven tools

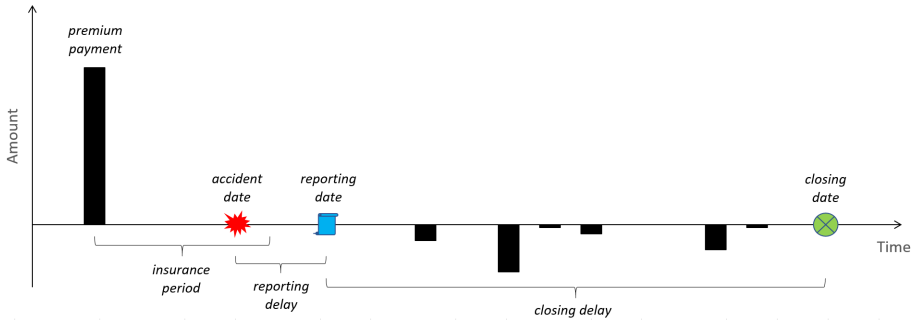


Fig. 3.2: Graphical representation of the claim timeline

are hardly adaptable to this problem. For this reason, we will not estimate the IBNYR reserve in our analysis, although it is a relevant topic for further development.
As soon as the accident is reported, the company is able to collect information about it, which represents the starting point for our analysis. Typically, the claim cannot be settled immediately for a number of reasons, including further investigation, new information, court decisions and so on. As a consequence, the claim is actually closed only at a future date $t_c \geq t_r$, the so-called *closing date*. The difference $\Delta := t_c - t_r$ is the *closing delay*, which may have very different features depending on the specific non-life business involved as well as the claim severity. For standard claims, it might be very small, like in health insurance contracts for the employees of a firm: the company receives standard claim documentation from the policyholder, approves it quickly, and refunds him/her with one single payment. On the other hand, more severe claims often lead to more tortuous - and longer - closing delay: sometimes no payment is due, sometimes a final payment is due, and sometimes company's investigation justifies claim benefits all the way along and a series of cash-flows is correspondingly paid. At reporting date, the company must allocate reserve to cover the payments expected during the closing delay. Such a reserve is the so-called *Reported-But-Not-Yet-Settled reserve* - or RBNYS reserve. Given that it is allocated at reporting date, when some information about the claim is already known, we can use it to build our individual reserve estimation.
The process described in this section is represented by the timeline in Figure 3.2.

3.3 Assumptions and model

Since our valuation date is the reporting date, we can assume that all the information about the claim is known at that date, and thus it can be used

to predict future payments. This is a first simplification, which may be unacceptable in some specific cases. In fact, one of the causes of the closing delay is the further investigation by the company, which could discover new information at a later date. For sake of simplicity, we will ignore this possibility.
According to the explanation in Section 3.2, the company pays an amount to the policyholder as soon as it is justified by the aforementioned investigation, thus generating a series of cash-flows. The emerging of such partial payments should be modelled, because it explains the change in reserve as the time passes. However, the dataset we will use includes the individual aggregated cost only, rather than the details on the cash-flows and the related timing. As a consequence, we will assume one single, aggregate payment for each claim due at closing date (nonetheless, repeated payments could potentially be discounted and summed to define one proper aggregate payment). Once these two assumptions are accepted, the model is a rather simple one. Assume that the closing delay Δ is a discrete variable measured in years, say $0, \dots, m$. That means: no payment is delayed more than m years after the reporting date. Moreover, consider a number of predictors $x_1, \dots, x_n$, that is, information about the policyholder, the claim, or any other relevant details. They are all available at reporting date, so they can be used to predict how likely the payment related to the claim i occurs after k years, that is, $\Delta_i = k$. Using a proper classification tool, formally represented by a function $p_k(\cdot)$ of the predictors, we will estimate $P(\Delta_i = k)$ for each admissible k:

$$\widehat{P}(\Delta_i = k) := p_k(x_{i1}, \dots, x_{in}), \qquad \forall k \in [0, m]. \tag{3.1}$$

Now, we can still use the same predictors $x_{i1}, \dots, x_{in}$ to estimate the payment amount due for the claim i; additionally, we will also use the information about Δ_i. Using a proper regression tool, formally represented by a function $f(x_{i1}, \dots, x_{in}|\Delta_i)$ of the closing delay and the predictors, we will estimate the payment amount:

$$\widehat{C}_i(\Delta_i) := f(x_{i1}, \dots, x_{in}|\Delta_i). \tag{3.2}$$

providing us with an estimation conditioned to Δ_i, which is unknown at reporting date. To overcome this limit, we first calculate the following estimations:

$$\widehat{C}_i(k) := f(x_{i1}, \dots, x_{in}|k), \qquad \forall k \in [0, m] \tag{3.3}$$

and then estimate the payment amount for the claim i as follows:

$$\begin{aligned} \widehat{C}_i &:= \sum_{k=0}^{m} \widehat{P}(\Delta_i = k)\widehat{C}_i(k) = \sum_{k=0}^{m} p_k(x_{i1}, \dots, x_{in}) f(x_{i1}, \dots, x_{in}|k) = \\ &= \sum_{k=0}^{m} p_k(\boldsymbol{x}_i) f(\boldsymbol{x}_i|k) \end{aligned} \tag{3.4}$$

which is an unconditioned estimation. Even if is not so common for case reserving to place such focus on settlement timing, it can be quite important when it comes with cash-flow management or any valuation based on discounted cash-flows (e.g., Solvency II).
Although such a theoretical model is quite general, the aforementioned assumptions are necessary to apply it on our specific data. In any case, if the reader wants to use it on its own available dataset and the final goal of its analysis, model and assumptions should be properly adapted.
In the next sections, we will present some of the machine learning techniques that can be used to estimate p_k and f. We will separate them because of their different nature: while p_k estimates the probabilities related to the categorical variable Δ_i, f estimates the claim amount $C_i(k)$. The former refers to a classification problem, whereas the latter refers to a regression problem.

3.4 Fundamental supervised learning tools

The closing delay estimation is a classification problem, that is, the goal is the prediction of how likely a claim will be closed - and the related amount paid - after k years from the reporting date. By contrast, the claim amount estimation is a regression problem since the target variable is numerical. The closing delay estimation could have been treated as a regression problem too, if we had assumed k as a continuous time variable, being inconsistent with the traditional assumption of a discrete k with some upper limit m, just like in any triangle-based reserve calculation exercise.
Although we will distinguish between tools used for closing delay estimation and tools used for claims amount estimation as described at the end of Section 3.1, this distinction is not strict. In fact, most of the fundamental machine learning tools we will recall in the following subsections - naïve Bayes, nearest neighbours, decision trees and neural networks - are flexible enough to be used for both classification and regression problems.

3.4.1 Naïve Bayes

This tool is surely one of the easiest machine learning techniques. It is a transformation of the well-known *Bayesian classifier*. Given the values for the predictor vector $\boldsymbol{x}_i$ related to the claim i, the Bayes' theorem returns the (exact) Bayesian classifier:

$$\begin{aligned} p_k(\boldsymbol{x}_i) &= P(\Delta_i = k|\boldsymbol{x} = \boldsymbol{x}_i) = \\ &= \frac{P(\Delta_i = k)P(\boldsymbol{x} = \boldsymbol{x}_i|\Delta_i = k)}{\sum_{h=0}^{m} P(\Delta_i = h)P(\boldsymbol{x} = \boldsymbol{x}_i|\Delta_i = h)}. \end{aligned} \tag{3.5}$$

This approach is theoretically correct, but presents a fundamental limit. The predictor in (3.5) implicitly assumes that we can find a sufficient num-

ber of records in the sample sharing the same vector $\boldsymbol{x}_i$. Perhaps, this is reasonable when there are VERY few predictors in the dataset, otherwise it is completely impracticable.
A straight modification to (3.5) represents a very simple solution to such a problem. If we give up to the assumption that the best probability estimation is only returned by those records matching the record to be classified, we will be able to use the whole dataset for the estimation. As a consequence of this assumption, the classifier in (3.5) changes as follows:

$$\begin{aligned} p_k(\boldsymbol{x}_i) &= P(\Delta_i = k|\boldsymbol{x} = \boldsymbol{x}_i) = \frac{P(\Delta_i = k)P(\boldsymbol{x} = \boldsymbol{x}_i|\Delta_i = k)}{\sum_{h=0}^{m} P(\Delta_i = h)P(\boldsymbol{x} = \boldsymbol{x}_i|\Delta_i = h)} = \\ &= \frac{P(\Delta_i = k)\prod_{j=1}^{n} P(x_j = x_{ij}|\Delta_i = k)}{\sum_{h=0}^{m} P(\Delta_i = h)\prod_{j=1}^{n} P(x_j = x_{ij}|\Delta_i = h)} \end{aligned} \tag{3.6}$$

which is, indeed, the so-called *naïve Bayes classifier*.
This approach is extremely simple to understand and to use. Moreover, it presents no computational issues: it is just a formula to apply as it is, rather than a complex algorithm. Unfortunately, this simplicity hides a major drawback, that is, the assumption of stochastic independence among predictors. In effect, that is exactly the assumption which allows us to move from the Bayesian classifier in (3.5) to the naïve Bayes classifier in (3.6). Comparing the two formulas, we can easily notice this point. However, this is seldom the case.
Moreover, some studies (for instance, Larsen (2005)) point out the strengths of such a classifier when it comes with record ranking, but also its weaknesses when it comes with probability estimation. In practice, naïve Bayes classifier's probability estimation can be very biased by the assumption of stochastic independence. If the same bias is shared by each record, the classification can be still good, but of course we cannot rely on the estimated probability. This is the reason why this classifier often outperforms more sophisticated classifiers as a classification tool, and it is still widely used in several fields (for instance, the spam filtering case in Shmueli et al. (2010)).
A last drawback is quite relevant. Actually, what if some predictor category is not present in the training dataset (for instance, it could be very rare)? In this case, $P(x_j = x_{ij}|\Delta_i = k) = 0$ for some j and k, thus $p_k(\boldsymbol{x}_i) = 0$, which is clearly wrong. In fact, the naïve Bayes classifier works well if each and every category is well represented. That has a two-fold meaning. First, the training dataset should be large enough to well represent each and every category. Second, more importantly, numerical predictors are not admissible by definition, that is, we can use it for closing delay estimation (which is a classification problem), but not for claim amount estimation (which is a regression model). For the latter, we need the methods described in the next subsections.

3.4.2 Nearest neighbours

The idea behind the nearest-neighbours algorithm is very intuitive, but it still guarantees a high level of adaptability to data. To score a new record, the method relies on finding the most "similar" records - the so-called "neighbours" - in the training dataset. In fact, this is a pure nonparametric method: no assumption needs to be established, no parameter needs to be estimated, no functional form needs to be assumed.
The sole issue regards the choice of a measure to calculate the "distance" between two records, that is, their grade of similarity. The most popular measure is the Euclidean distance defined in (2.14). It is worth noting that this definition of distance would give much more weight to higher scales than lower scales, so all the predictors should be first standardized before computing (2.14). Otherwise, the nearest-neighbours algorithm could result in extremely biased predictions.
Once we have chosen a distance measure, we can calculate the distance between any pair of records, but we still need a rule to score new records. The simplest rule is: a new record is classified in the same category of its closest neighbour. Therefore, given the predictors of such a record, we will compute all the distances between it and the records in the training dataset. Among them, pick up that with the smallest distance to the new record: its category will be assigned to the new record itself. In effect, we have just applied a 1-nearest neighbours algorithm.
Nonetheless, the approach might be easily generalized to any number k of neighbours. Instead of picking up the closest record only, pick up the k closest records, and assign the majority class among them to the new record. In practice, the usage of more neighbours tends to reduce misclassification error. If we use one neighbour only, it could be the case that a record is the closest one to the new record only by chance: there could be noise there, rather than information. To some extent, the greater the k, the lower the error, the greater the predictive power. On the other hand, however, if k is too high, we will miss out on the method's ability to capture the local structure in the data, that is, we will ignore information.
In any case, the choice of k is straightforward. First, choose an upper limit K for k. Then, score each record in the *validation* dataset using the closest record in the training dataset (1-nearest neighbours algorithm), and calculate the validation error ϵ_1. Likewise, score each record in the *validation* dataset using the two closest records in the training dataset (2-nearest neighbours algorithm), and calculate the validation error ϵ_2. Repeat the process until the nearest-neighbours algorithm, that is, the last admissible algorithm according to our upper limit. At the end, we will get the validation errors $\epsilon_1, \ldots, \epsilon_K$. Finally, pick up the k related to the smallest validation error, say k_{min}, and use the k_{min}-nearest neighbours algorithm for scoring.
Remember that ϵ_k denotes the validation error of nearest-neighbours algo-

rithm. Correspondingly, we could compute the training error as well, that is, the error committed when scoring each record in the *training* (rather than validation) dataset using the k closest records in the training dataset itself. The training error, however, cannot be used for scoring because the smallest training error always relates to the 1-nearest neighbours algorithm: whichever the training record to be scored, its closest training record is obviously the record itself!
Of course, the nearest-neighbours classifier is as simple as the naïve Bayes classifier. As already discussed at the beginning of this section, its main advantage is the lack of parametric assumptions. Unfortunately, there are some drawbacks. For instance, from a computational perspective, this algorithm can be very expensive. Sometimes, data scientists try to reduce predictors by using other, less expensive machine learning tools before applying the nearest-neighbours algorithm to their datasets. Dimension reduction my be performed, among others, by classification trees, which is the topic of the next subsection.

3.4.3 Classification and regression trees (CARTs)

Decision trees were used as a machine learning tool in Breiman et al. (1984) for the very first time to segment a population by splitting up the dataset through binary rules. The algorithm is now referred to as *classification tree*. Since one of our goals is the categorical classification among the closing delay classes, this tool is a good candidate for us. By contrast, we should properly adapt it to regression problems, if we want to use it to predict the claim amount. In this case, we call it a *regression tree*. However, given that the basic algorithm does not change, we can always refer to *classification and regression trees*, or CARTs.
CARTs are based on recursive partitioning, which divides up the multidimensional space (that is, the dataset) of the explanatory variables into non-overlapping multidimensional rectangles. This division is accomplished recursively, that is, operating on the results of the prior divisions. An example of CART is in Figure 3.3. First, one of the explanatory variables is selected, say $x_{k(0)}$ (the first *node* of the tree, so-called *root*), and a value of $x_{k(0)}$, say $s_{k(0)}$), is chosen to split the n-dimensional space into two parts: one part contains all the records with $x_{k(0)} \leq s_{k(0)}$, say $n(1,1)$ records, while the other with all the records with $x_{k(0)} > s_{k(0)}$, say $n(1,2)$ records. The two subsets represent the first level of the tree. Let's consider one of them: it could be either *pure* (i.e., it contains only records sharing the same value of the independent variable) or *impure*. In the first case, no further split is possible, so the subset will represent a *leaf* of the tree. In the second case, other splits are possible, so the subset will represent another node of the tree. In Figure 3.3, leaves are denoted by green rectangles, while nodes are denoted by blue circles. Unless both of the subsets generated by the root

node are pure, one of them (at least) will be divided in a similar manner by choosing a variable again and a split value for the variable. In Figure 3.3, where both of the subsets of the first level are impure, they represent the two nodes of the first level, and the initial dataset is further partitioned into four regions:

- the first one contains the $n(2,1)$ records with $x_{k(1,1)} \leq s_{k(1,1)}$, and represents a node for the next level using $x_{k(2,1)}$ as a splitting variable
- the second one contains the $n(2,2)$ records with $x_{k(1,1)} > s_{k(1,1)}$, and represents a node for the next level using $x_{k(2,2)}$ as a splitting variable
- the third one contains the $n(2,3)$ records with $x_{k(1,2)} \leq s_{k(1,2)}$, and represents a leaf for the next level containing all the records with target variable equal to $y_{h(2,3)}$
- the fourth one contains the $n(2,4)$ records with $x_{k(1,2)} > s_{k(1,2)}$, and represents a node for the next level using $x_{k(2,4)}$ as a splitting variable.

Since some splits are still possible, the recursive partitioning goes on, getting smaller and smaller subsets, either nodes or leaves. Sooner or later, we will have divided the whole dataset up into pure subsets (of course, this is not always possible, as there may be records that belong to different classes but have exactly the same values for everyone of the predictor variables).
In the case of closing delay estimation, the dataset will be partitioned into subsets which contain either claims closed in the reporting year (closing delay 0), or claims closed the year after (closing delay 1), or claims closed two years after (closing delay 2), ..., or claims closed m years after (closing delay m). In fact, the classification tree resulting from recursive partitioning is a *pure* tree: all the closing delay categories are perfectly separated.
The main problem of recursive partitioning is the choice of the splitting rule node by node, that is, the choice of $x_{k(\cdot,\cdot)}$ and $s_{k(\cdot,\cdot)}$ at each step of the algorithm. Assume to define an *impurity function* $i(A)$ as an impurity measure of some rectangle A, or its related node. A specific splitting rule on A results in two subrectangles A_L and A_R, which are generally impure, that is, $i(A_L)$ and $i(A_R)$ are both nonzero. Intuitively, we want to choose the splitting rule in order to minimize some combination of $i(A_L)$ and $i(A_R)$. The most natural choice is the function

$$I(A_L, A_R) := \frac{card(A_L)}{card(A)} i(A_L) + \frac{card(A_R)}{card(A)} i(A_R) \tag{3.7}$$

which is the average of the two impurity measures, weighted by the number of observations in each rectangle, that is, its cardinality. By comparing the reduction in $I(A_L, A_R)$ across all possible splits in all possible predictors, the next split is chosen.

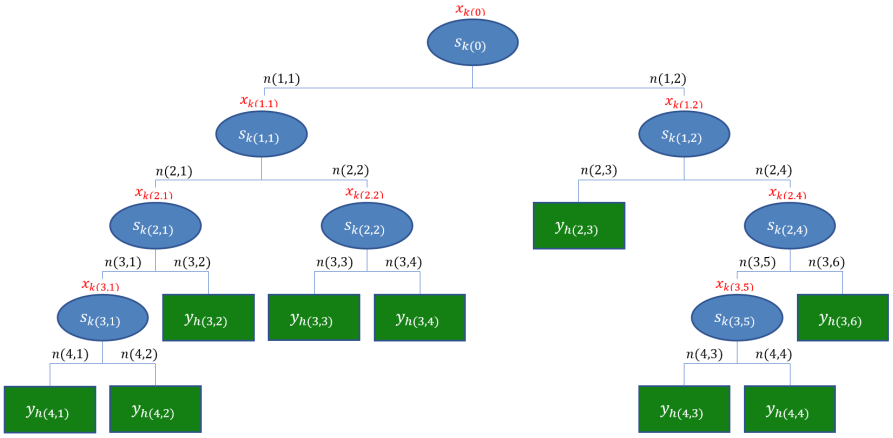


Fig. 3.3: Example of CART produced by recursive partitioning

What about the impurity function i? In our application and in most of them, one uses the *Gini index* (as defined in Shmueli et al. (2010)):

$$i(A) := 1 - \sum_{k=0}^{m} p_k^2(A) \tag{3.8}$$

where p_k is the proportion of records in rectangle A that are closed after k year from the reporting. However, other impurity measures are also widely used, for example the *entropy index* (as defined in Shmueli et al. (2010)):

$$E(A) := -\sum_{k=0}^{m} p_k(A) \log_2[p_k(A)]. \tag{3.9}$$

All in all, so far the algorithm is quite intuitive as well as its application in classifying new records. For instance a new observation, whose explanatory values are known, will be dropped down the tree until it reaches a leaf. So the new observation will be simply classified on the base of the specific leaf's classification.
As discussed at the beginning of the section, the algorithm can be easily adapted to predict numerical variables. We only need some changes. First, the response value assigned to a record in a leaf is determined by the average of the response variable among the records in that leaf (by contrast, in a classification tree, it is determined by one of the possible category of the response variable). Second, given that we cannot use discrete measure of impurity such as the Gini index and the entropy index, the typical impurity measure used in regression problems is the sum of squared deviations from the mean, that is, the sum of squared errors.
By definition, recursive partitioning produces a tree which classify the records

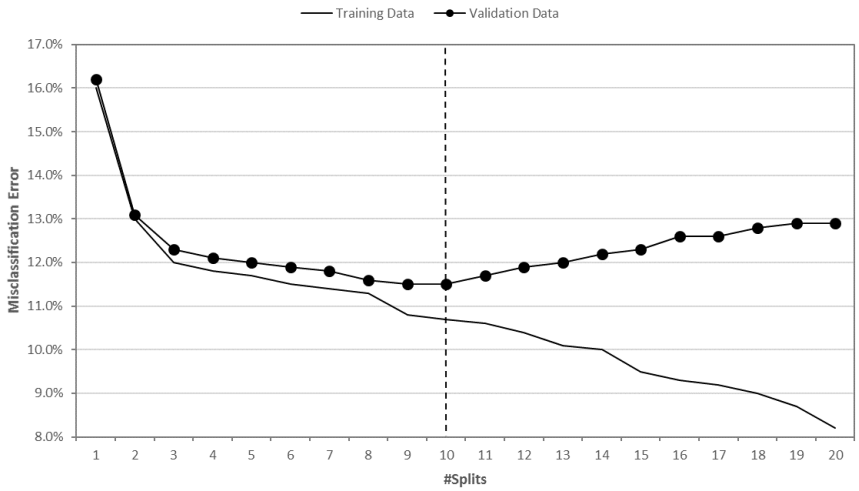


Fig. 3.4: Misclassification error and minimum-error model selection

without errors. Actually, we used a training dataset to train the classification tree, which perfectly predict closing delay on that dataset. But what if we use the same tree on the validation dataset? In general, the predicted values on the validation dataset will result in a positive *misclassification error*. In fact, the error cannot be zero on datasets other than the training dataset itself. However, our full classification tree has a major drawback represented by Figure 3.4. As it usually happens through the first splits on the validation dataset, the full tree can still guarantee comparable misclassification errors on the two datasets. However, as the number of splits increases, the full tree starts *overfitting* the validation data: since it fully reflects the training dataset without distinguishing between "signal" and "noise", the noisy component cause too high misclassification error in the validation dataset. Indeed, the typical consequence of overfitting is that, after some number of splits, the misclassification error on the validation dataset stops decreasing and starts increasing (in Figure 3.4, it occurs after ten splits). In the first ten splits both the training and validation misclassification errors decrease, but thereafter the full tree overfits the validation data.
Overfitting prevents us from using the full tree for predicting purposes, so we need to choose another tree, that is, some subtree of the full tree. There are several criteria to do that, but two major categories of methods can be distinguished:

- forward stopping-tree methods
- backward pruning-tree methods.

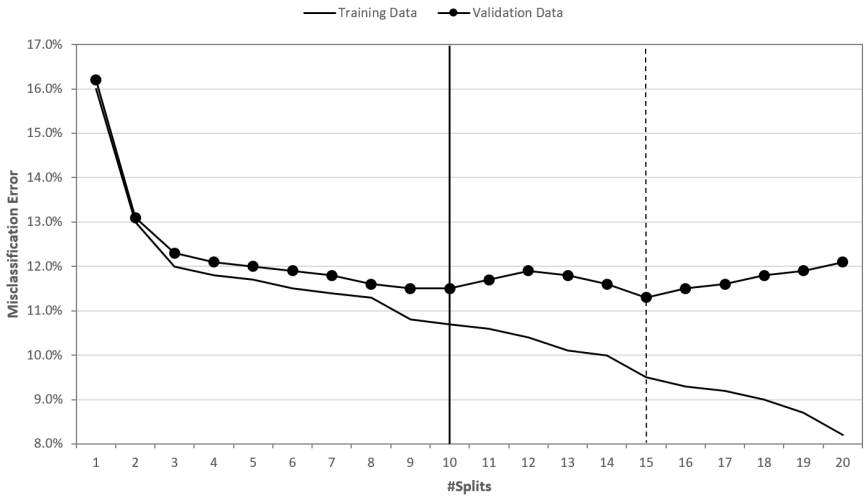


Fig. 3.5: Misclassification error and best-pruning model selection

Some empirical forward methods can be easily implemented by setting conditions to the tree characteristics such as maximum number of splits, minimum number of records in a node and minimum reduction in impurity. However, these approaches are merely based on the tree complexity, rather than its predictive power.

Among the forward methods, the most natural choice is suggested by the Figure 3.4 itself, that is, we can simply use the tree consisting of the first n splits that do not induce overfitting ($n = 10$ in Figure 3.4). In other words, we let the full tree grow until the first step leading to a validation error higher than the previous step. If we have new observations to classify, they will be dropped down this subtree until they reach a leaf.

More complex methods have been developed as well, for instance, the so-called *chi-squared automatic in-training data* (CHAID). At each node, the algorithm select the predictor with the strongest association to the target variable, measured by the p-value of the chi-squared test of independence. If such a p-value is low enough, the split of the node significantly improves its purity, so the algorithm will carry it out, and the growth of the tree goes on. Otherwise, the growth is stopped.

The alternative to stopping the growth of the tree is represented by pruning the tree, that is, "climb" the training full tree and "chop" the weakest branches until some conditions are met. Intuitively, pruning the tree is more computationally expensive, because we have to build the full tree in any case, and then work on that further. However, it has been proven to be more successful too, and it is not so surprising. In effect, the tree is pruned considering more information: not only what-if-we-keep-the-smaller-tree in-

formation, but also what-if-we-take-the-bigger-tree information.
A simple backward approach follows the idea of picking the tree with the smallest validation error, just like in one of the forward methods. However, the full tree is still built until the last leaf, and then we choose the subtree leading to the smallest validation error. Notice that we would still pick the subtree after 10 splits in Figure 3.4, but that is not always the case (see, for instance, Figure 3.5).
However, choosing the subtree according to the smallest validation error only means we completely ignore the complexity of the tree. For instance, take a look at Figure 3.5 once again. We are picking the 15-split subtree since it leads to the smallest error, but we get a rather small error with the 10-split subtree as well. Actually, we accept five splits more - a relevant increase in complexity - for a little decrease in error. Somehow, it does not seem to be the best choice. To consider this issue in pruning the tree, we may use the so-called *cost complexity criterion* as described in Breiman et al. (1984). For a tree with L leaves and training error ϵ, the cost complexity is defined as follows:

$$\gamma := \alpha L + \epsilon \tag{3.10}$$

where α denotes a (nonnegative) penalty factor for the tree size as measured by the number of leaves. Notice that, if $\alpha = 0$, there is no penalty for the tree size, and the best tree is simply the full tree itself. On the other hand, the greater the α, the greater the relevance of the tree size in pruning the tree. If α is great enough, the training error is no longer relevant for the algorithm, and the tree is pruned until the very first node, that is, the root. The algorithm therefore starts with the n-level full tree, and compares its γ with the γ of all the possible subtrees with $n-1$ levels. Starting from $\alpha = 0$, the α gets increased little by little, until the γ of the full tree exceeds that of one of those subtrees. Such a subtree is considered as the best of its level, and the same procedure is repeated starting from it. In fact, the algorithm finds the best subtree of each size based on the cost complexity criterion: the lower γ, the better the subtree. In practice, we get a sequence of "best subtrees" for their sizes, based on the training data only. Finally, the so-called *best pruned tree* used for scoring will be the subtree among them leading to the smallest validation error.
All in all, the full tree is useless for scoring purposes, rather it is just the formal result of recursive partitioning. What is really useful for prediction is the subtree extracted by using one of the several algorithms to stop growing or pruning the full tree.
So far we described the fundamentals of CARTs. One of the reasons for their popularity is that they are adaptable to a wide variety of applications, and have been successfully used in many situations. In particular, if there is a highly non-linear and complex relationship to describe, decision trees may outperform regression models. Furthermore, CARTs do not require

massive data preparation, that is, they can handle non-standardized data, categorical data, missing data, outliers and so on. By contrast, we should standardize the variables and take the natural logarithm of some numerical variables before running standard linear regression, logistic regression, or even the nearest-neighbours method described in Subsection 3.4.2. Finally, trees provide easily understandable classification rules (at least if they are not too large), even easier than in regression.
An important advantage of CARTs is that no further selection algorithm is necessary. As opposed to parametric regression methods, the process itself selects the most relevant explanatory variables. We simply let the machine learning tool run on the whole dataset, and the resulting tree will include only some of the explanatory variables, which are the most significant on the base of the impurity measure used to split the dataset.

3.4.4 Neural networks

Algorithms like nearest-neighbours method or CARTs have the major advantage of non-dependence on underlying structures or parameters. This is what makes them perfect to handle - and, to some extent, discover - unknown relationships in datasets. On the other hand, this flexibility makes them extremely weak when data is not enough to train them properly. *Neural networks* are trained by data too, but assume an underlying function that is generally much more complex than the typical functions used in regression. To some extent, we might consider neural networks as a trade-off between pure nonparametric methods and traditional regressions.
A number of successful applications contributed to the great spread of the neural network concept, including some relevant financial topics (see Trippi et al. (1996)) such as bankruptcy prediction, asset allocation, fraud detection and customer relationship management.
An example of neural network is in Figure 3.6. Actually, each neural network has its own structure based on *neurons*, but all of them share the same fundamental features:

- one *input layer* consisting of a number of neurons - one for each predictor
- one or more *hidden layers*, each of them consisting of its own neurons
- one *output layer* consisting of a number of neurons which returns the predictions.

In particular, notice that the output layer consists of one neuron only if the target variable is binary (i.e., the neural network predicts one probability only) or numerical (i.e., the neural network predicts one numerical value only).

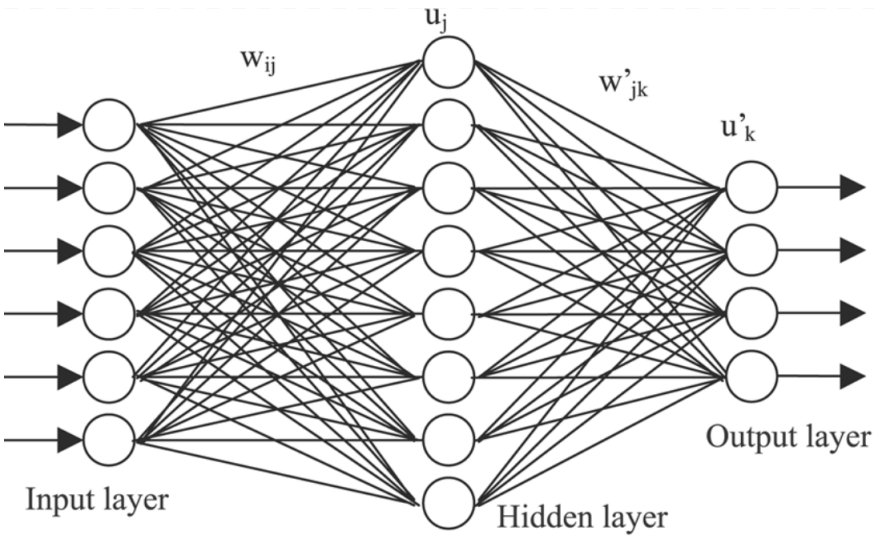


Fig. 3.6: Example of neural network

Another important remark regards the number of hidden layers. One issue within this subject on which there is a consensus is the performance difference from adding additional hidden layers: the situations in which performance improves with a second (or third, etc.) hidden layer are very few. One hidden layer is sufficient for the large majority of problems, as stated in Shmueli et al. (2010) as well:

> *The most popular choice for the number of hidden layers is one. A single hidden layer is usually sufficient to capture even very complex relationships between the predictors.*

This is an empirical statement coming from the evidence of a large number of applications with neural networks. More importantly, it has been given a rigorous theoretical foundation in Hornik et al. (1989), where the authors prove that one-hidden-layer neural networks can approximate any deterministic measurable function arbitrarily well, as long as sufficiently many hidden neurons are available. Therefore, we will only consider one-hidden-layer neural networks, just like the example in Figure 3.6.
Furthermore, each input neuron is connected to each hidden neuron, and each hidden neuron is connected to each output neuron.
Neural network training is based on the calibration of the following parameters (see Figure 3.6):

- the weight parameters w_{ij}, one for each connection from the input layer to the hidden layer
- the bias parameters u_j, one for each hidden neuron

- the weight parameters w'_{jk}, one for each connection from the hidden layer to the output layer
- the bias parameters u'_k, one for each output neuron.

The input layer, which knows the raw data of the predictors x_i, communicate it to the hidden layer. Such an information, however, is weighted by the connection itself, that is, by the related weight parameter. In other words, the j^{th} hidden neuron receives the information $w_{ij}x_i$. Additionally, given that it receives data from each and every input neuron (i.e., predictor) at the same time, it aggregates information through some *hidden activation function* $f(\boldsymbol{x}, \boldsymbol{w}_j, u_j)$ of the predictor values, the weights and the bias. The most popular hidden activation function is the traditional weighted average:

$$H_j := f(\boldsymbol{x}, \boldsymbol{w}_j, u_j) = u_j + \sum_i w_{ij}x_i, \qquad \forall j. \tag{3.11}$$

Likewise, the output neurons receive weighted information from each hidden neuron. In particular, the k^{th} output neuron receives the value $w'_{jk}H_j$ from the j^{th} hidden neuron. In fact, the k^{th} output neuron receives information from each and every hidden neuron, so it will manipulate it too. More specifically, the output neurons use an *output activation function* $g(\boldsymbol{H}, \boldsymbol{w}'_k, u'_k)$ of their own parameters.
As we have already mentioned, Hornik et al. (1989) proves that one-hidden-layer neural networks can approximate any deterministic measurable function to any degree of accuracy. Surprisingly, this statement still holds regardless of the activation function we use, as long as it is nonconstant and continuous. Therefore, the choice of g is completely up to the data scientist, and may be adapted to the specific application. However, there are some properties that make an activation function preferable to others:

- nonlinearity, which allows for the detection of nonlinear relationships in the dataset
- continuous differentiability, which guarantees the convergence through gradient-based optimization methods (e.g., the back propagation we will describe later)
- monotonicity, which makes the error surface convex, guaranteeing one single global minimum and avoiding any issue related to possible local minimums (see Wu (2009)).

As a result, the range of activation functions that guarantee convergence and stability in any neural network is quite reduced. Here is a list of the most common activation functions sharing the three aforementioned properties:

$$O_k \quad := \quad \text{logit}\left(u'_k + \sum_j w'_{jk}H_j\right) = \frac{1}{1 + e^{-u'_k - \sum_j w'_{jk}H_j}} \tag{3.12}$$

$$O_k := \tanh\left(u'_k + \sum_j w'_{jk} H_j\right) = \frac{2}{1 + e^{-2\left(u'_k + \sum_j w'_{jk} H_j\right)}} - 1 \quad (3.13)$$

$$O_k := \arctan\left(u'_k + \sum_j w'_{jk} H_j\right) = \tan^{-1}\left(u'_k + \sum_j w'_{jk} H_j\right) \quad (3.14)$$

$$O_k := \text{softsign}\left(u'_k + \sum_j w'_{jk} H_j\right) = \frac{u'_k + \sum_j w'_{jk} H_j}{1 + \left|u'_k + \sum_j w'_{jk} H_j\right|} \quad (3.15)$$

$$O_k := \text{ISRU}\left(u'_k + \sum_j w'_{jk} H_j\right) = \frac{u'_k + \sum_j w'_{jk} H_j}{\sqrt{1 + \alpha\left(u'_k + \sum_j w'_{jk} H_j\right)^2}} \quad (3.16)$$

where $\alpha > 0$ in the inverse-square-root-unit (ISRU) function. O_k always represents the prediction provided by the k^{th} output neuron. If the target variable is categorical, O_k equals the probability of the k^{th} category related to a specific record. All in all, as long as those properties hold, the choice of the activation function into the standard architecture of a neural network (Figure 3.6) is important to the extent it makes the convergence slower or faster. Indeed, the neural network will anyway converge to the same solution - the global minimum - regardless of the activation function. Throughout this dissertation, we will always use the logit activation function in 3.12.
There is still an interesting observation to do. Assume that the target variable is binary, that is, the neural network gets only one output neuron that predicts the probability of "success", whereas the number of hidden neuron is the same as the number of predictors. Moreover, assume the following parameters within the hidden layer:

$$u_i = 0, \qquad w_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}, \qquad \forall i, j \quad (3.17)$$

which actually means

$$H_j \equiv H_i = x_i, \qquad \forall j. \quad (3.18)$$

Using the logit activation function in (3.12) for the single output neuron, we get

$$\widehat{P}(1) = O_1 := g(\boldsymbol{x}, \boldsymbol{w}'_k, u'_k) = \text{logit}\left(u'_1 + \sum_i w'_{i1} x_i\right) = \frac{1}{1 + e^{-u'_1 - \sum_i w'_{i1} x_i}} \quad (3.19)$$

which is equivalent to the functional form of a logistic regression's prediction. However, we are going to see why it does not mean that this peculiar neural network equals the logistic regression in terms of predicted probabilities.
In fact, the main difference between neural networks and regressions lies in the way parameters are estimated. While regression methods rely on

predetermined target functions to minimize or maximize, neural networks actually "learn" from data. Estimating biases and weights is a consequence of such a learning process, whichever its algorithm is. The most popular one is the so-called *back propagation*, which we will describe now.
We denoted the prediction from the k^{th} node by O_k. Moreover, let's define the prediction error related to the first record as

$$\epsilon_{1k} := O_{1k}(1 - O_{1k})(y_1 - O_{1k}) \tag{3.20}$$

where y_1 equals the actual value of the target variable. Given a global *learning rate* $\lambda \in (0, 1)$ and some initialization values for the parameters of the network weights and biases are updated as follows, for each i and j:

$$\begin{aligned} w_{ij} &\longrightarrow w_{ij} + \lambda\epsilon_{1k} && (3.21)\\ u_j &\longrightarrow u_j + \lambda\epsilon_{1k} && (3.22)\\ w'_{jk} &\longrightarrow w'_{jk} + \lambda\epsilon_{1k} && (3.23)\\ u'_k &\longrightarrow u'_k + \lambda\epsilon_{1k}. && (3.24) \end{aligned}$$

After that, the second record goes through the network to get its own estimations O_{2k}, then the error is computed:

$$\epsilon_{2k} := O_{2k}(1 - O_{2k})(y_2 - O_{2k}) \tag{3.25}$$

and the parameters are updated once again starting from those of the previous step:

$$\begin{aligned} w_{ij} &\longrightarrow w_{ij} + \lambda\epsilon_{2k} && (3.26)\\ u_j &\longrightarrow u_j + \lambda\epsilon_{2k} && (3.27)\\ w'_{jk} &\longrightarrow w'_{jk} + \lambda\epsilon_{2k} && (3.28)\\ u'_k &\longrightarrow u'_k + \lambda\epsilon_{2k}. && (3.29) \end{aligned}$$

The computation is repeated for all the records, all the way through the training dataset: after the last observation, the first *epoch* is completed. Generally, a number of epochs is predefined, that is, the records are estimated several times until some tolerance on the significance of parameter updating is broken, or some threshold on the training error is finally met.
What we have just described is called *case updating*, but it is not the sole option. For instance, in *batch updating*, the whole training dataset is run through the network before each updating takes place. As a consequence, the parameters are updated on the base of the overall training error, or its average:

$$\bar{\epsilon}_k := \frac{1}{N}\sum_i \epsilon_{ik} \tag{3.30}$$

and, for each i and j,

$$w_{ij} \longrightarrow w_{ij} + \lambda\bar{\epsilon}_k \tag{3.31}$$
$$u_j \longrightarrow u_j + \lambda\bar{\epsilon}_k \tag{3.32}$$
$$w'_{jk} \longrightarrow w'_{jk} + \lambda\bar{\epsilon}_k \tag{3.33}$$
$$u'_k \longrightarrow u'_k + \lambda\bar{\epsilon}_k. \tag{3.34}$$

In practice, case updating tends to be more accurate than batch updating, but it is also more computationally expensive, given that the parameters are updated N times per epoch rather than only once.
As long as nonlinear, differentiable and monotonic activation functions are chosen for the neural network, meta-parameters such as the learning rate and the number of epochs are relevant to the extent they facilitate convergence to the global minimum, but they do not affect the overall predictive performance of the algorithm. For this reason, given that the logit activation function is the only one used in our applications, we will arbitrarily set them to $\lambda = 0,1$ and $N = 2000$ respectively, while the initial values of bias and weights are all zero.
Neural networks can be very powerful, if their architecture is significant, that is, the number of hidden layers and hidden neurons is "right" - whatever it means. There are algorithms that automatically select this features, but none of them seems clearly superior to a simple trial-and-error approach (see Shmueli et al. (2010)). Nonetheless, network architecture depends on the predictors too. Unfortunately, neural network are rigid in this sense: they cannot really choose among predictors, as opposite to other methods such as stepwise regressions and CARTs. Neural networks always use all the predictors given as input, so they should be chosen very carefully by the data scientist, for instance, by using a proper selection method. Clustering, PCA and CARTs themselves are all suitable approaches.
The various forms of their architecture give neural network a unique flexibility in dealing with data. Potentially, they can recognize any type of pattern. However, the architecture itself is the origin of their major drawback too, that is, their black-box structure. While anyone can easily "read" a tree, or interpret the parameters of a regression, this is generally impossible when dealing with neural networks. Of course, knowing the transfer functions and any parameters, we may write the ultimate function returning predictions, but then we would probably find no meaning in that. Too many parameters and too complex functions are often involved in neural network, and it must be accepted as it is. The usual validation tools can be used to measure the predictive power of a neural network, but unfortunately we cannot rely on model interpretation.

3.5 An application to Australian bodily injury data

As we may have noticed in Section 3.4, machine learning tools are very flexible, and could potentially improve many of the traditional processes in actuarial practice. Non-life reserving is just one of them.
This section describes the path that has been followed to predict the closing delay and claim amount on a publicly available motor insurance dataset.

3.5.1 Data

The data comes from an R package containing a number of datasets for actuarial applications (see Dutang et al. (2016)). Additionally, it was also used in De Jong et al. (2008) as a starting point for generalized regression analysis. In Dutang et al. (2016), the dataset is called *Automobile bodily injury claim dataset in Australia* (`ausautoBI8999`), with the following description:

> *This dataset contains information on 22.036 settled personal injury insurance claims in Australia. These claims arose from accidents occurring from July 1989 through to January 1999. Claims settled with zero payment are not included.*

Notice that it only includes the claims that were settled as at January 1999, and all of them are supposed to be definitely closed, although some might have been reopened afterwards. Actually, the dataset relates to a small part of the entire claim scope of a motor insurance company, but it is probably one of the most interesting parts to us. Bodily injury claims are indeed the most expensive and long-lasting ones, thus some of the most important claims to analyse individually. Moreover, the dataset does not include any information on timing and amount of partial payments, so triangle-based methods do not represent a viable solution. For these reasons, case reserving seems to be a natural choice in this specific situation.
The dataset includes the following fields:

- `AccDate` for the claim accident date
- `ReportDate` for the claim reporting date
- `FinDate` for the claim closing date date
- `AccMth` for the claim accident month
- `ReportMth` for the claim reporting month
- `FinMth` for the claim closing month
- `OpTime` for the total time it took to close the claim (*operational time*)
- `InjType1` for the injury severity of the first injured person

- `InjType2` for the injury severity of the second injured person
- `InjType3` for the injury severity of the third injured person
- `InjType4` for the injury severity of the fourth injured person
- `InjType5` for the injury severity of the fifth injured person
- `InjNb` for the number of injured persons (max 5)
- `Legal` for the legal representation (yes/no)
- `AggClaim` for the aggregate claim amount after closing.

The injury severity variables are categorical, but we converted them in numerical between 0 and 100, in order to calculate an overall severity score for each claim. The conversion is based on the rules in Figure 3.7, and the overall score of a single claim is given by the sum of its own scores (notice that it will be always positive because each claim caused one injury at least). Therefore, we define the numerical variables `InjScore1`, `InjScore2`, `InjScore3`, `InjScore4` and `InjScore5`, in correspondence to the original `InjType1`, `InjType2`, `InjType3`, `InjType4` and `InjType5`. Additionally, we get the overall severity score variable `InjScoreTot`. It is worth mentioning that, in many jurisdictions, care costs for seriously injured individuals could outstrip compensation costs for the fatal claims. In other words, assigning the latter to the highest injury score (see Figure 3.7) could potentially lead to bias. However, we performed some checks in the dataset to prove that this is not the case. For instance, comparing the claims with one single severe injury to the claims with one single fatal injury, we observe that the average of the former is materially lower than the average of the latter.
From the variables `AccDate`, `ReportDate` and `FinDate`, we extract the year - `AccYr`, `ReportYr` and `FinYr` - and use it to add two more variables:

- `ReportTime` for the reporting delay defined as `ReportYr` − `AccYr`
- `FinTime` for the closing delay defined as `FinYr` − `ReportYr`.

One last additional variable that will be useful to evaluate linear correlations is `LnAggClaim`, that is, the natural logarithm of `AggClaim`.
Using `ReportYr` and `FinTime`, let's have a look at the run-off triangle of the entire dataset in Figure 3.8 for the claim numbers, in Figure 3.9 for the claim payments and in Figure 3.10 for the claim average payments (a dozen of claims relating to the highest amounts have been excluded). Apparently, some reporting years and closing delays include too few observations, as we can easily observe in Figure 3.8. This is the reason why we will select a proper training/validation dataset as well as a test dataset for the ultimate analysis. More specifically, the reporting years 1993-1996 for the closing

category	**score**
NA (no injury)	0
minor injury	25
small injury	25
medium injury	50
not recorded	50
high injury	75
severe injury	75
fatal injury	100

Fig. 3.7: Codification of injury severity

reporting	*closing delay*						
year	0	1	2	3	4	5	6
1993	**1.191**	**1.970**	**1.152**	**558**	426	277	49
1994	**874**	**1.531**	**870**	**709**	443	65	0
1995	**718**	**1.381**	**1.188**	**881**	134	0	0
1996	**529**	**1.511**	**1.314**	**174**	0	0	0
1997	585	2.020	283	0	0	0	0
1998	807	352	0	0	0	0	0
1999	14	0	0	0	0	0	0

Fig. 3.8: Claim numbers

reporting	*closing delay*						
year	0	1	2	3	4	5	6
1993	**26.038.265**	**77.236.474**	**74.285.051**	**58.268.484**	48.169.135	38.347.205	8.444.935
1994	**16.746.725**	**42.454.730**	**42.485.823**	**55.798.479**	44.833.960	7.119.057	0
1995	**6.076.308**	**20.958.156**	**41.895.020**	**50.580.334**	8.286.925	0	0
1996	**4.090.537**	**21.665.535**	**46.736.025**	**8.158.885**	0	0	0
1997	3.787.223	33.675.352	8.543.935	0	0	0	0
1998	6.207.963	3.475.759	0	0	0	0	0
1999	36.599	0	0	0	0	0	0

Fig. 3.9: Claim amounts

reporting	*closing delay*						
year	0	1	2	3	4	5	6
1993	**21.863**	**39.206**	**64.484**	**104.424**	113.073	138.438	172.346
1994	**19.161**	**27.730**	**48.834**	**78.700**	101.205	109.524	0
1995	**8.463**	**15.176**	**35.265**	**57.412**	61.843	0	0
1996	**7.733**	**14.339**	**35.568**	**46.890**	0	0	0
1997	6.474	16.671	30.191	0	0	0	0
1998	7.693	9.874	0	0	0	0	0
1999	2.614	0	0	0	0	0	0

Fig. 3.10: Claim average amounts

delays from 0 to 3 (in bold in Figure 3.8, 3.9 and 3.10) and will be considered for training and validation, while the reporting years 1997-1998 will be considered for test using all the available closing delays, that is, from 0-2 for 1997 and 0-1 for 1998 (in Roman in Figure 3.8, 3.9 and 3.10). All the other data (in grey in Figure 3.8, 3.9 and 3.10) are not necessary for the analysis, and will be excluded.
Figure 3.8 also demonstrate that there is no regular path by reporting year or closing delay. For instance, the zero-delay claims tend to decrease by reporting year, while the 3-year-delay claims have a rather different path. Comparing reporting years, we see less differences, but still some relevant discordances. For instance, compare the 1-year-delay claim average payment to the 2-year-delay claim average payment in Figure 3.10: in 1993 the latter is about 50% greater than the former, while in 1996 the latter is about 150% greater than the former.
To better include timing information, we consider both the reporting year and the closing delay as categorical variables. As a consequence, we convert each of them in the three corresponding binary variables (the fourth one would be redundant):

- `ReportYr` is converted to `ReportYr1` for 1994, `ReportYr2` for 1995 and `ReportYr3` for 1996
- `FinTime` is converted to `FinTime1` for one year, `FinTime2` for two year and `FinTime3` for three year.

Similarly, we convert `InjNb` to `InjNb2` for two injuries, `InjNb3` for three injuries, `InjNb4` for four injuries and `InjNb5` for five injuries.
Finally, we will use the following fields: `InjScoreTot`, `InjNb2`, `InjNb3`, `InjNb4`, `InjNb5`, `LegalBin`, `ReportTime`, `ReportYr1`, `ReportYr2`, `ReportYr3`, `FinTime`, `FinTime1`, `FinTime2`, `FinTime3`, `AggClaim` and `LnAggClaim` (besides an ID variable to identify the different records).

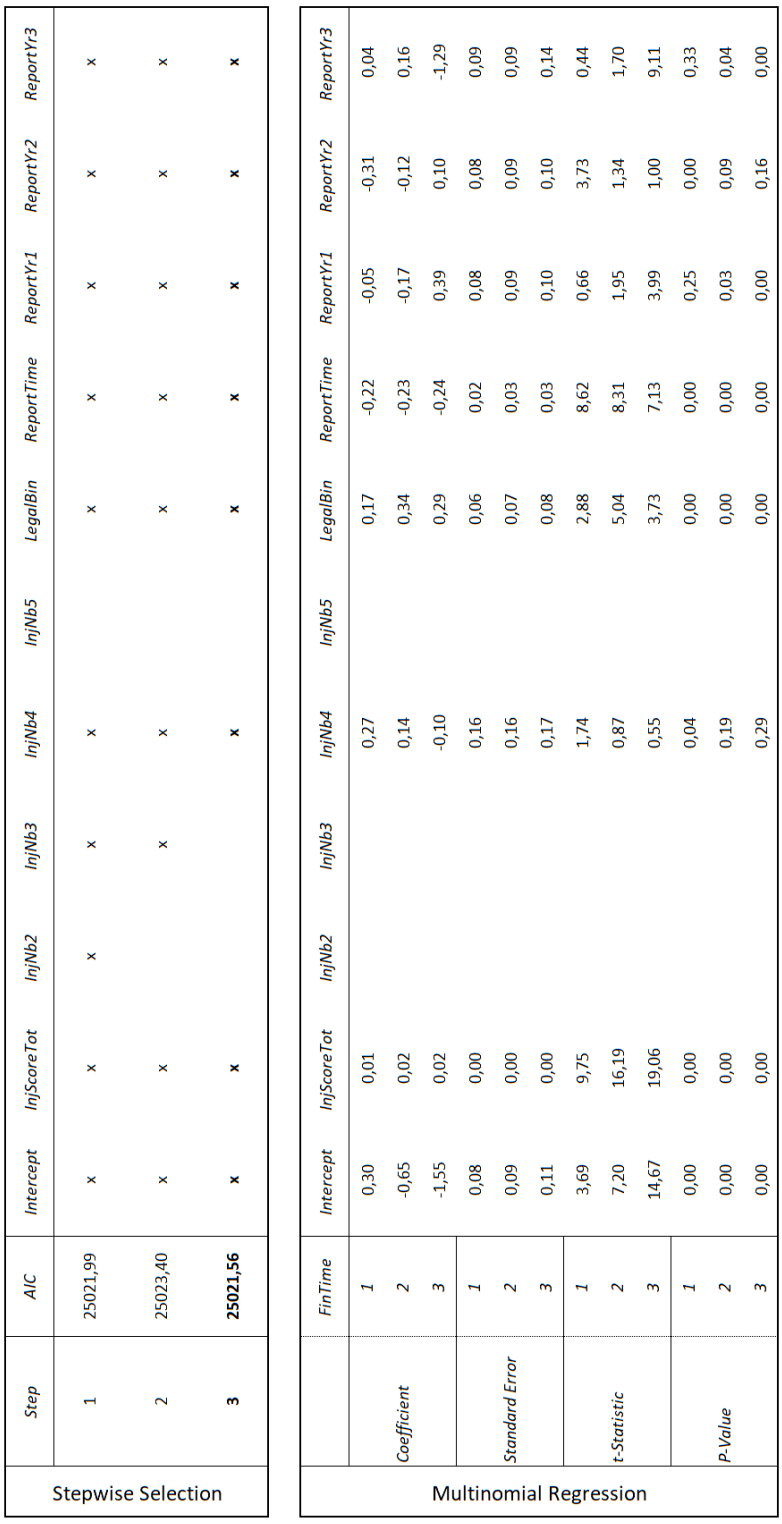
3.5.2 Claim closing delay estimation

In our framework, estimating the closing delay means using the predictors `InjScoreTot`, `InjNb2`, `InjNb3`, `InjNb4`, `InjNb5`, `LegalBin`, `ReportTime`, `ReportYr1`, `ReportYr2` and `ReportYr3` as inputs for some machine learning tool to return an estimation of the probability that the claim is definitely closed after 0, 1, 2 and 3 years after the reporting. In other words, the target variable is `FinTime`. We use four methods: multinomial regression, naïve Bayes, nearest neighbours and classification tree.
The first method we will consider is the *multinomial regression*, that is, a generalization of the logistic regression for count variables with more than two categories. It is a GLM whose target variable is distributed as a multinomial on a discrete and finite domain: in our case, it will be the range

	InjScoreTot	InjNb2	InjNb3	InjNb4	InjNb5	LegalBin	ReportTime	ReportYr1	ReportYr2	ReportYr3	FinTime1	FinTime2	FinTime3	LnAggClaim	AggClaim
mean	56,99	0,19	0,13	0,08	0,09	0,63	0,82	0,24	0,25	0,21	0,39	0,27	0,14	**9,59**	**35.857,34**
st. deviation	37,98	0,39	0,34	0,27	0,29	0,48	1,14	0,43	0,43	0,41	0,49	0,45	0,35	**1,42**	**67.073,67**
skewness	1,36	1,58	2,15	3,14	2,79	-0,52	1,39	1,21	1,14	1,40	0,47	1,02	2,07	**-0,53**	**5,42**
kurtosis	2,16	0,49	2,63	7,86	5,79	-1,73	1,26	-0,53	-0,69	-0,04	-1,78	-0,97	2,29	**1,48**	**42,21**

	InjScoreTot	InjNb2	InjNb3	InjNb4	InjNb5	LegalBin	ReportTime	ReportYr1	ReportYr2	ReportYr3	FinTime1	FinTime2	FinTime3	LnAggClaim
InjScoreTot	100,0%													
InjNb2	-5,4%	100,0%												
InjNb3	22,3%	-19,0%	100,0%											
InjNb4	36,9%	-14,1%	-11,5%	100,0%										
InjNb5	68,6%	-15,6%	-12,6%	-9,4%	100,0%									
LegalBin	9,1%	0,1%	2,8%	4,1%	6,6%	100,0%								
ReportTime	-13,0%	-7,8%	-9,5%	-7,1%	-8,8%	8,3%	100,0%							
ReportYr1	-8,7%	1,0%	-4,2%	-4,4%	-6,2%	-29,4%	2,2%	100,0%						
ReportYr2	17,9%	6,4%	10,0%	8,9%	10,0%	3,2%	-20,3%	-32,7%	100,0%					
ReportYr3	17,8%	4,9%	8,2%	8,2%	11,0%	19,0%	-20,5%	-29,3%	-30,2%	100,0%				
FinTime1	-6,9%	1,7%	-0,1%	-0,6%	-6,7%	-0,1%	-2,1%	-0,2%	-6,5%	4,5%	100,0%			
FinTime2	13,0%	1,6%	4,3%	5,4%	9,2%	5,6%	-6,9%	-6,9%	1,5%	11,6%	-48,7%	100,0%		
FinTime3	14,6%	-0,7%	2,2%	3,2%	11,7%	-1,2%	-3,1%	6,1%	11,9%	-13,6%	-32,0%	-24,8%	100,0%	
LnAggClaim	**26,8%**	**-1,4%**	**3,9%**	**7,2%**	**19,5%**	**20,6%**	**27,7%**	**0,9%**	**-5,1%**	**-10,4%**	**-11,9%**	**20,7%**	**28,1%**	**100,0%**

Fig. 3.11: Descriptive statistics of closing delay's and claim amount's explanatory variables



Stepwise Selection

Step	AIC	Intercept	InjScoreTot	InjNb2	InjNb3	InjNb4	InjNb5	LegalBin	ReportTime	ReportYr1	ReportYr2	ReportYr3
1	25021,99	x	x	x	x	x		x	x	x	x	x
2	25023,40	x	x		x	x		x	x	x	x	x
3	**25021,56**	x	x			x		x	x	x	x	x

Multinomial Regression

	FinTime	Intercept	InjScoreTot	InjNb2	InjNb3	InjNb4	InjNb5	LegalBin	ReportTime	ReportYr1	ReportYr2	ReportYr3
Coefficient	1	0,30	0,01			0,27		0,17	-0,22	-0,05	-0,31	0,04
	2	-0,65	0,02			0,14		0,34	-0,23	-0,17	-0,12	0,16
	3	-1,55	0,02			-0,10		0,29	-0,24	0,39	0,10	-1,29
Standard Error	1	0,08	0,00			0,16		0,06	0,02	0,08	0,08	0,09
	2	0,09	0,00			0,16		0,07	0,03	0,09	0,09	0,09
	3	0,11	0,00			0,17		0,08	0,03	0,10	0,10	0,14
t-Statistic	1	3,69	9,75			1,74		2,88	8,62	0,66	3,73	0,44
	2	7,20	16,19			0,87		5,04	8,31	1,95	1,34	1,70
	3	14,67	19,06			0,55		3,73	7,13	3,99	1,00	9,11
P-Value	1	0,00	0,00			0,04		0,00	0,00	0,25	0,00	0,33
	2	0,00	0,00			0,19		0,00	0,00	0,03	0,09	0,04
	3	0,00	0,00			0,29		0,00	0,00	0,00	0,16	0,00

Fig. 3.12: Regression summary for closing delay estimation after stepwise selection

closing delay	prior probability
0	20,01%
1	38,86%
2	27,00%
3	14,14%

Fig. 3.13: Prior probabilities using training data

$\{0, 1, 2, 3\}$.
Dealing with regressions, we should pay attention to possible multicollinearities that could distort the results. Figure 3.11 reports the correlation matrix of the dataset. If all the correlations among the binary variables of the same categorical variable are excluded, few remaining correlations are significant. Therefore, we will run the multinomial regression including all the predictors except for `InjNb5`, whose correlation with `InjScoreTot` is the highest one, equal to 68,6%. Figure 3.12 reports the regression summary; in particular, observe that `InjNb2` and `InjNb3` have been dropped out by the stepwise algorithm (given that the software does not have an automatic selection algorithm for multinomial regressions, we did it manually, because the explanatory variables are relatively few).
Moreover, using the definition of naïve Bayes classifier in 3.6, we can easily apply the naïve Bayes method, which is quite straightforward and does not need further discussion.
After that, let's perform the same estimation by using the nearest-neighbours method. According to the description in Subsection 3.4.2, the number k of neighbours should correspond to the lowest validation error. As demonstrated in Figure 3.18, it occurs when $k = 19$. It is worth noting the peculiar, opposite shape of the two lines. In Subsection 3.4.2, we noticed that the training dataset has actually nothing to gain from nearest-neighbours prediction: in fact, the more the neighbours, the greater the error. This happens because the best prediction in the training dataset always occurs when $k = 1$ by definition. On the contrary, the validation error slightly decreases as k increases since the algorithm neglects more and more noisy information. So this is not surprising if the greatest validation error occurs when $k = 1$, just as the training error is minimum.
Likewise, we need to choose a proper classification tree on the base of the validation error as well. In Figure 3.21, we observe that the lowest validation error occurs after twelve splits (grey dashed line in Figure 3.21), but, as discussed in Subsection 3.4.3, this is not necessarily the best choice if we want to take into account the complexity of the tree too. This is the reason why we will use the best-pruned tree for scoring, that is, the tree

Actual	Predicted Class				Cases	Errors	Errors
Class	0	1	2	3	Number	Number	Percentage
0	97	1.805	73	12	1.987	1.890	95,12%
1	130	3.339	329	61	3.859	520	13,47%
2	83	2.082	445	71	2.681	2.236	83,40%
3	44	1.027	207	126	1.404	1.278	91,03%
Total					**9.931**	**5.924**	**59,65%**

Fig. 3.14: Multinomial regression summary results using training data

Actual	Predicted Class				Cases	Errors	Errors
Class	0	1	2	3	Number	Number	Percentage
0	72	1.194	44	15	1.325	1.253	94,57%
1	92	2.185	220	37	2.534	349	13,77%
2	52	1.444	287	60	1.843	1.556	84,43%
3	29	664	155	70	918	848	92,37%
Total					**6.620**	**4.006**	**60,51%**

Fig. 3.15: Multinomial regression summary results using validation data

consisting of three splits (black dashed line in Figure 3.21). Furthermore, Figure 3.22, 3.23 and 3.24 shows the full tree, the minimum error tree and the best pruned tree respectively.
Remember: we are not really interested in classifying records among the four classes, rather we will directly use the estimated probabilities. However, trying to classifying them using the various methods is the easiest way to compare their performances. Therefore, such an assessment is reported in Figure 3.14-3.15, 3.16-3.17, 3.19-3.20 and 3.25-3.26. There we can observe the related confusion matrices together with the training and validation errors. All the errors seems to swing around 60%. In other words: if we use the rule that assigns a record the greatest predicted probability among the four possible category, we will correctly predict around 40% closing delays. Actually, that represents a rather poor result, and it can be proven through a straightforward remark, the so-called *naïve rule*. The training data is characterized by the prior (empirical) probabilities per closing delay category as in Figure 3.13. The most obvious prediction algorithm would classify all the records in the category that appears most often in the training dataset, that is, category 1 with 38,86%. In such a case, we would correctly predict 38,86% claims, that is, an error of 61,14%. But it is just slightly higher than the overall validation errors of the multinomial regression (see Figure 3.15), the nearest-neighbours algorithm (see Figure 3.20) and the best pruned classification tree (see Figure 3.26). Curiously, it is even lower than the overall

Actual	Predicted Class				Cases	Errors	Errors
Class	0	1	2	3	Number	Number	Percentage
0	740	1.069	151	27	1.987	1.247	62,76%
1	870	2.210	689	90	3.859	1.649	42,73%
2	434	1.380	764	103	2.681	1.917	71,50%
3	242	730	281	151	1.404	1.253	89,25%
Total					**9.931**	**6.066**	**61,08%**

Fig. 3.16: Naïve Bayes summary results using training data

Actual	Predicted Class				Cases	Errors	Errors
Class	0	1	2	3	Number	Number	Percentage
0	477	708	122	18	1.325	848	64,00%
1	536	1.454	490	54	2.534	1.080	42,62%
2	284	988	483	88	1.843	1.360	73,79%
3	153	445	226	94	918	824	89,76%
Total					**6.620**	**4.112**	**62,11%**

Fig. 3.17: Naïve Bayes summary results using validation data

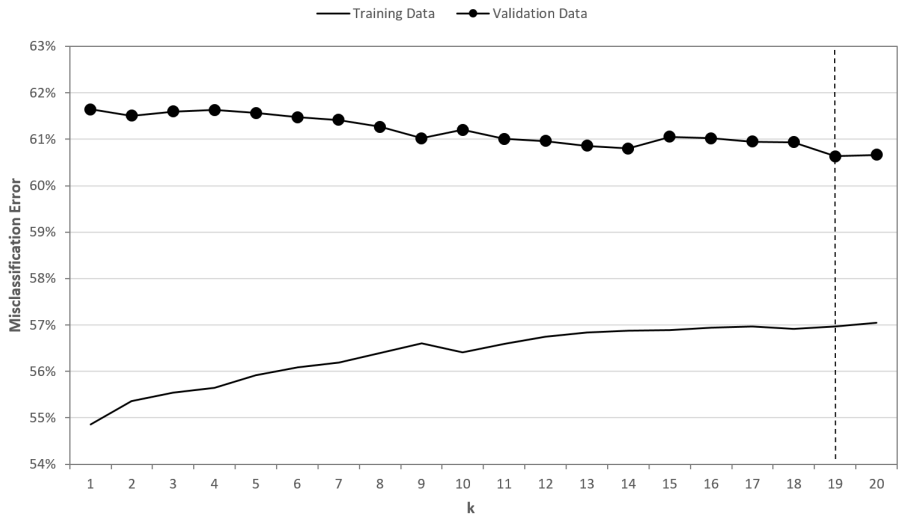


Fig. 3.18: Nearest neighbours - training and validation error varying by k

Actual	Predicted Class				Cases	Errors	Errors
Class	0	1	2	3	Number	Number	Percentage
0	472	1.389	105	21	1.987	1.515	76,25%
1	367	3.140	286	66	3.859	719	18,63%
2	222	1.869	502	88	2.681	2.179	81,28%
3	115	938	191	160	1.404	1.244	88,60%
Total					**9.931**	**5.657**	**56,96%**

Fig. 3.19: Nearest neighbours summary results using training data

Actual	Predicted Class				Cases	Errors	Errors
Class	0	1	2	3	Number	Number	Percentage
0	267	959	83	16	1.325	1.058	79,85%
1	269	1.990	224	51	2.534	544	21,47%
2	164	1.355	248	76	1.843	1.595	86,54%
3	75	614	128	101	918	817	89,00%
Total					**6.620**	**4.014**	**60,63%**

Fig. 3.20: Nearest neighbours summary results using validation data

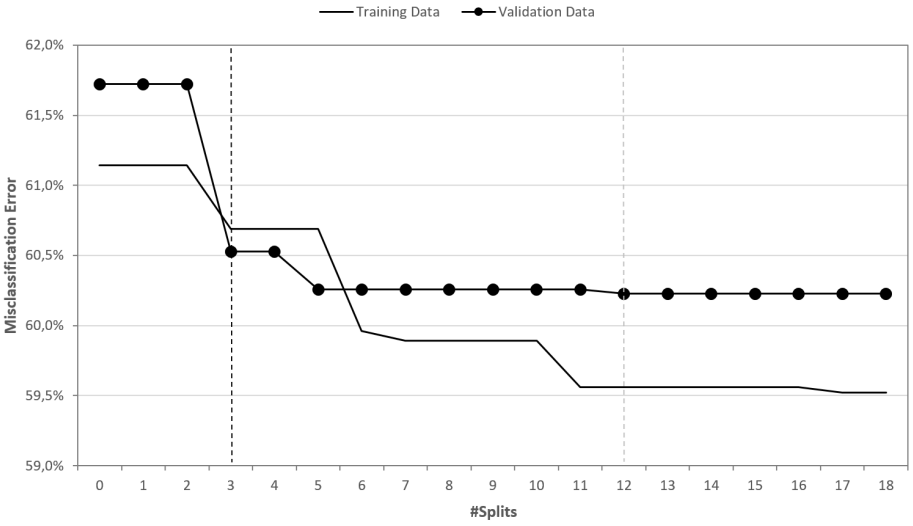


Fig. 3.21: Classification tree - training and validation error varying by number of splits

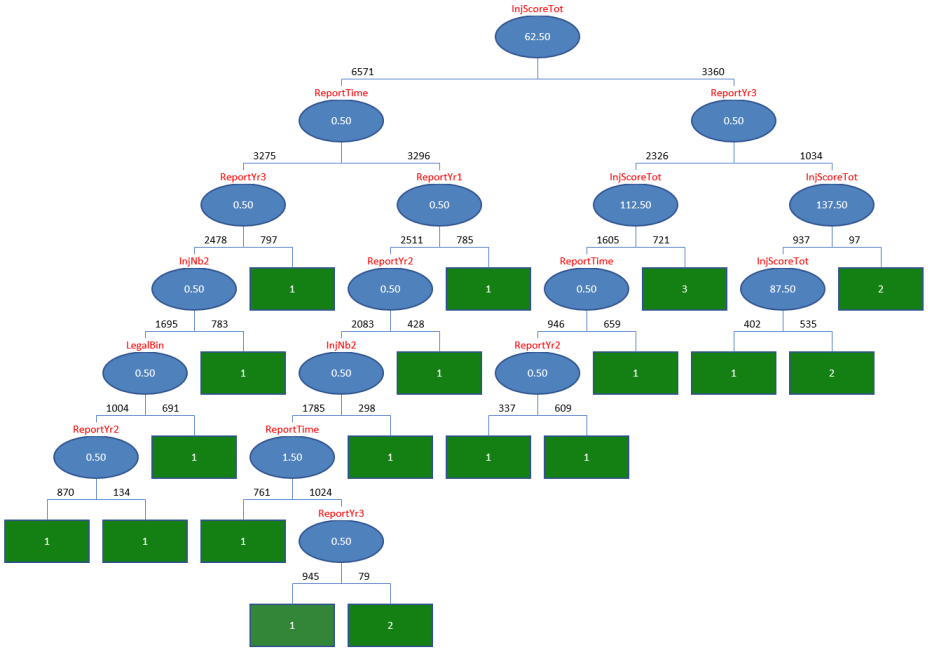


Fig. 3.22: Full classification tree

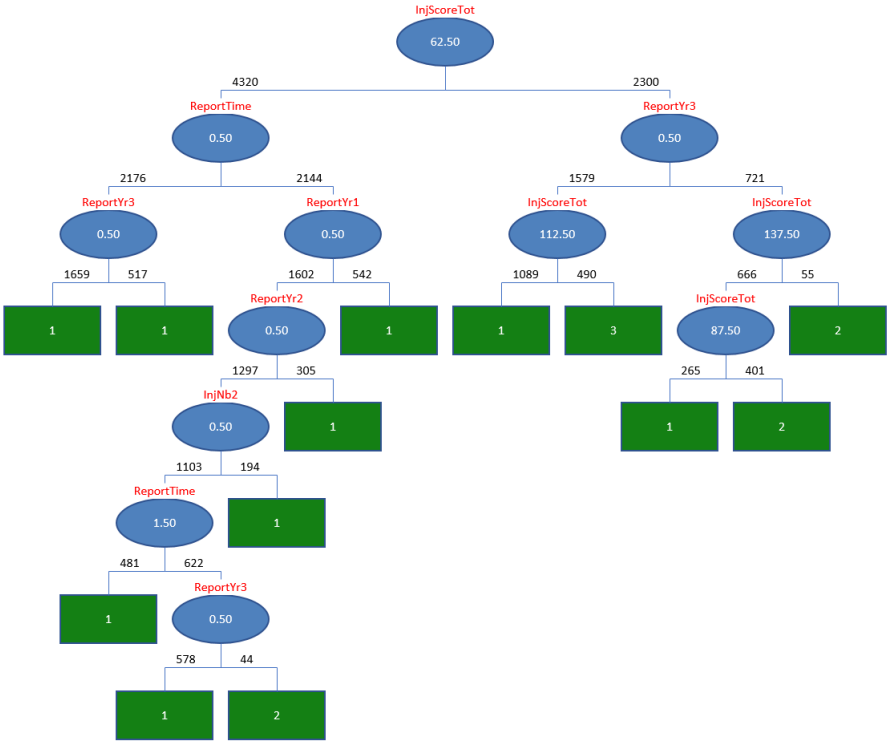


Fig. 3.23: Minimum error classification tree

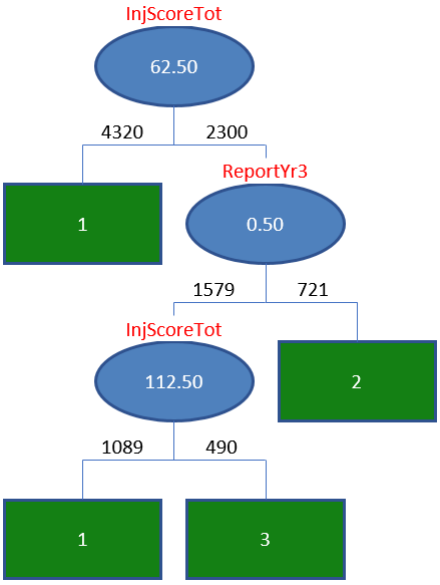


Fig. 3.24: Best pruned classification tree

Actual	*Predicted Class*				*Cases*	*Errors*	*Errors*
Class	0	1	2	3	*Number*	*Number*	*Percentage*
0	0	1.892	57	38	1.987	1.987	100,00%
1	0	3.412	258	189	3.859	447	11,58%
2	0	2.101	347	233	2.681	2.334	87,06%
3	0	1.094	49	261	1.404	1.143	81,41%
Total					**9.931**	**5.911**	**59,52%**

Fig. 3.25: Classification tree summary results using training data

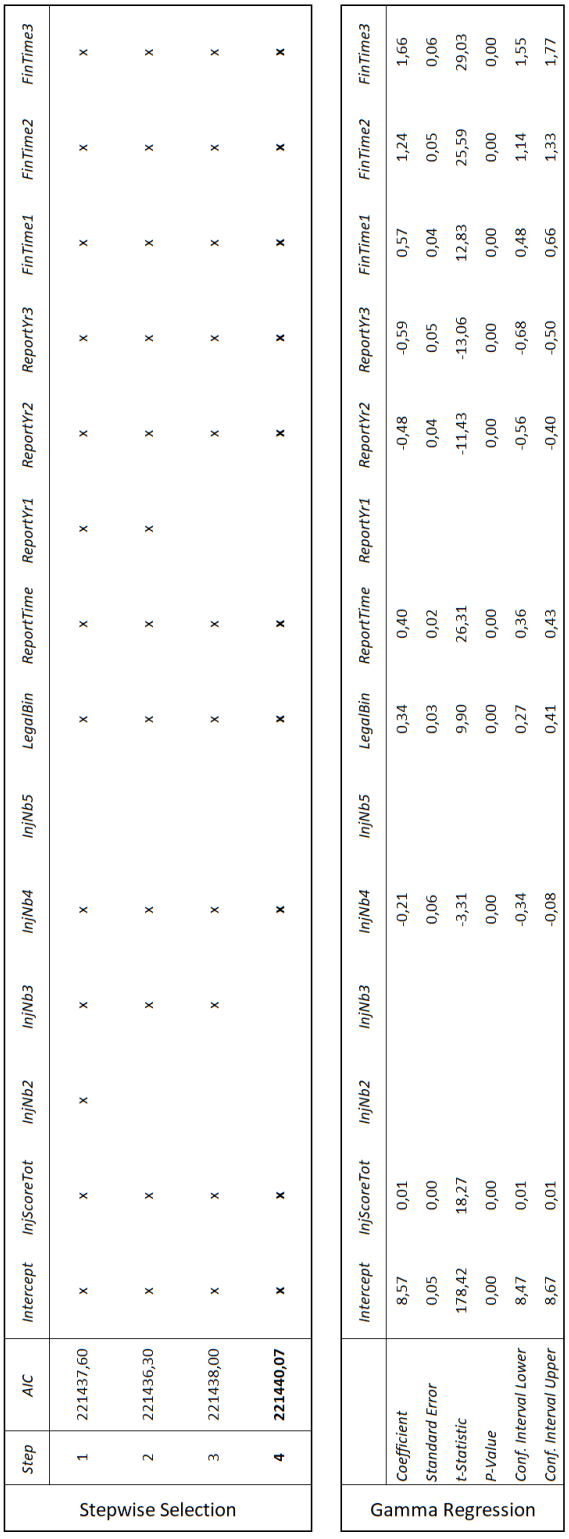
Actual	*Predicted Class*				*Cases*	*Errors*	*Errors*
Class	0	1	2	3	*Number*	*Number*	*Percentage*
0	0	1.229	67	29	1.325	1.325	100,00%
1	0	2.121	294	119	2.534	413	16,30%
2	0	1.363	315	165	1.843	1.528	82,91%
3	0	696	45	177	918	741	80,72%
Total					**6.620**	**4.007**	**60,53%**

Fig. 3.26: Classification tree summary results using validation data

validation error of naïve Bayes (see Figure 3.17). In other word, it does not seem we gain any insight by using machine learning on this dataset to predict closing delay. This is simply due to the low informative value of the data itself. This is not always the case. For instance, as we will observe later on, some machine learning methods will return more accurate claim amount estimations as compared to traditional regression methods.

3.5.3 Claim payment amount estimation

In our framework, estimating the claim payment amount means using the predictors previously used to estimate the closing delay together with the binary variables `FinTime1` for the 1-year closing delay, `FinTime2` for the 2-year closing delay and `FinTime3` for the 3-year closing delay. They will represent the inputs for machine learning tools to return estimations of `AggClaim`. We use three methods: gamma regression, regression tree and neural network.
When it comes with regression methods with numerical fields, we should pay attention to the features of our predictors. First, it's important to reduce asymmetries and too heavy tails. This can be checked by having a look at the skewness and kurtosis in Figure 3.11. Only the target variable `AggClaim` got very high skewness (5,42) and kurtosis (42,21), but this is simply due to the nature of the variable itself. A simple solution is represented by the natural logarithm of `AggClaim`, `LnAggClaim`, which would permit us to use multiple linear regression (Figure 3.11 also reports skewness and kurtosis of `LnAggClaim`: both of them are very low). However, it would also introduce a significant component of transformation bias (see Subsection 1.6.1), so we will not try this approach. Rather, we will use gamma regression, a typical choice in non-life actuarial practice (for instance, see Charpentier (2015b)) to overcome the typical obstacles in amount estimations.
Another typical issue related to regression models is multicollinearity among explanatory variables. However, as already done in Subsection 3.5.2, we will run the regression including all the predictors except for `InjNb5`, whose correlation with `InjScoreTot` is the highest one, equal to 68,6% (see Figure 3.11). Figure 3.27 shows the related results. By comparing Figure 3.11 and Figure 3.27, it is worth noting that the selected 10-coefficient model includes the predictors characterized by the highest correlations with `LnAggClaim`. Moreover, the exclusion of most of the `InjNb` binary variables leads to the exclusion of some among the most relevant multicollinearities, that is, those among `InjNb` and `InjScoreTot` (see Figure 3.11). Actually, this is not surprising: to some extent, in effect, the greater the number of injuries, the greater the overall claim severity.
When it comes with regression trees, we need to do some further remarks with respect to classification tree. Actually, the former predict numerical variables, while the latter classify records among a range of categories. Given that each claim relates to a different payment, a mere full tree would get



Stepwise Selection

Step	AIC	Intercept	InjScoreTot	InjNb2	InjNb3	InjNb4	InjNb5	LegalBin	ReportTime	ReportYr1	ReportYr2	ReportYr3	FinTime1	FinTime2	FinTime3
1	221437,60	x	x	x	x	x		x	x	x	x	x	x	x	x
2	221436,30	x	x		x	x		x	x	x	x	x	x	x	x
3	221438,00	x	x		x	x		x	x		x	x	x	x	x
4	**221440,07**	**x**	**x**			**x**		**x**	**x**		**x**	**x**	**x**	**x**	**x**

Gamma Regression

	Intercept	InjScoreTot	InjNb2	InjNb3	InjNb4	InjNb5	LegalBin	ReportTime	ReportYr1	ReportYr2	ReportYr3	FinTime1	FinTime2	FinTime3
Coefficient	8,57	0,01			-0,21		0,34	0,40		-0,48	-0,59	0,57	1,24	1,66
Standard Error	0,05	0,00			0,06		0,03	0,02		0,04	0,05	0,04	0,05	0,06
t-Statistic	178,42	18,27			-3,31		9,90	26,31		-11,43	-13,06	12,83	25,59	29,03
P-Value	0,00	0,00			0,00		0,00	0,00		0,00	0,00	0,00	0,00	0,00
Conf. Interval Lower	8,47	0,01			-0,34		0,27	0,36		-0,56	-0,68	0,48	1,14	1,55
Conf. Interval Upper	8,67	0,01			-0,08		0,41	0,43		-0,40	-0,50	0,66	1,33	1,77

Fig. 3.27: Regression summary for claim amount estimation after stepwise selection

minimum records per leaf	average training error	average validation error	average overall error	percentage weighted error	percentage overall error
10	28.262	28.554	28.379	16,63%	0,38%
20	28.393	28.622	28.485	16,74%	0,38%
30	27.819	28.020	27.899	14,28%	0,39%
40	27.819	28.020	27.899	14,28%	0,39%
50	**26.695**	**27.528**	**27.028**	**3,67%**	**0,29%**
60	26.982	27.477	27.180	5,06%	0,22%
70	27.088	27.326	27.183	5,54%	0,10%
80	27.154	27.393	27.250	5,49%	0,09%
90	27.185	27.331	27.243	6,10%	0,01%
100	27.273	27.385	27.318	6,30%	0,15%
110	27.339	27.457	27.386	6,83%	0,16%
120	27.525	27.568	27.542	9,14%	0,16%
130	27.603	27.596	27.600	9,47%	0,10%
140	27.647	27.679	27.660	9,83%	0,25%
150	27.729	27.753	27.739	9,85%	0,41%
160	27.632	27.680	27.651	8,43%	0,37%
170	27.652	27.662	27.656	8,45%	0,40%
180	27.743	27.718	27.733	9,09%	0,39%
190	27.764	27.741	27.755	9,38%	0,43%
200	27.806	27.797	27.802	9,60%	0,51%
210	27.830	27.777	27.809	9,59%	0,48%
220	27.842	27.788	27.820	9,81%	0,50%
230	27.826	27.766	27.802	9,67%	0,48%
240	27.952	27.866	27.918	9,93%	0,54%
250	27.971	27.906	27.945	10,18%	0,55%
260	27.971	27.913	27.948	10,96%	0,56%
270	28.000	28.116	28.046	11,02%	0,67%
280	28.012	28.139	28.063	11,30%	0,72%
290	28.032	28.171	28.088	11,50%	0,69%
300	28.196	28.430	28.289	13,86%	0,63%

Fig. 3.28: Regression tree performance varying by minimum number of records per leaf

as many leaves as the number of records in the training dataset. Obviously, this is not feasible, so we need an additional rule to stop the growth of the full tree, and the prediction related to a specific leaf will equal the average payment of the training records in that leaf. Generally, the rule is quite empirical, for instance, a maximum number of tree levels, a maximum number of nodes and so on.

In our analysis, we choose a minimum number of records in any leaf, and then predictions and errors from that tree are evaluated. The results are summarized in Figure 3.28, not only training error, validation error and overall error, but also a percentage weighted error and a percentage overall error. The percentage weighted error is based on the sixteen percentage

errors per reporting year and closing delay weighted on the actual payment amount itself. Instead, the percentage overall error is simply the percentage difference between the total of the payments in the dataset and the total of the related estimates.

First, Figure 3.28 shows no overfitting, which is a quite important advantage for any predictive model. In other words, training error and validation error are very close regardless of the minimum records per leaf. Secondly, observe the fluctuation of the percentage errors in the last to columns in Figure 3.28. When very few records are required in each leaf, it seems that a lot of noise affects the tree: errors are quite material, especially the percentage weighted errors, which are greatest between 10 and 40 minimum records per leaf. After all, the regression tree is predicting much better between 50 and 100: it will be probably there where it performs at best. Nonetheless, further increase in minimum records per leaf implies a new increase in error: in fact, if too many records are required into each leaf, the regression tree will no longer be able to detect information in data.

All in all, the chosen tree is highlighted in bold in Figure 3.28, that is, the tree leading to the lowest percentage weighted record 3,67%. Further, although it leads to the lowest average overall error, it does not lead to the lowest percentage overall error (which is however very low).

Unfortunately, we cannot report the whole full tree and best pruned tree since they are too big (the former got 15 levels, while the latter got 13 levels). However, Figure 3.29 reports the best pruned tree in tabular form from the root to the fifth level. Although most of the predictors already appear there, the most important ones are those used for splitting in the very first levels. They are likely reducing error the most. The injury score and the reporting time have a primary importance since they define the root and the first level of the tree respectively. The binary variable for the 3-year closing delay is also quite relevant, given that it separates almost all the records on the second level. This is not a surprise, if you look back at the correlations in Figure 3.11: `InjScoreTot`, `ReportTime` and `FinTime3` correspond to the greatest correlations with the claim amount. However, observe that `InjNb2` and `ReportYr1` are used on some nodes in the fourth level, although they are the least correlated predictors with the claim amount in Figure 3.11. It is probably indicating that the tree is trying to explain variability that cannot be detected by the logistic regression, which has indeed excluded those variables through the stepwise routine (see Figure 3.27).

As briefly mentioned in Subsection 3.4.4, standard neural networks lack an embedded algorithm to select relevant predictors and exclude irrelevant predictors, but we may refer to the predictors implicitly selected by the regression tree itself. Since it has used all of them (few are used in deeper levels of the tree, so they do not appear in Figure 3.29), we will run the neural network on the full dataset.

So far, we know the structure of the input layer: thirteen neurons for thir-

Level	Item ID	Item Type	Parent ID	Left Child ID	Right Child ID	Split Variable	Split Value	Cases	Prediction
0	0	Node	-	1	2	InjScoreTot	137,5	6620	36.078
1	1	Node	0	3	4	ReportTime	1,5	6420	33.046
1	2	Node	0	5	6	ReportTime	1,5	200	134.457
2	3	Node	1	7	8	FinTime3	0,5	5032	24.782
2	4	Node	1	9	10	FinTime3	0,5	1388	61.671
2	5	Node	2	11	12	FinTime3	0,5	154	103.858
2	6	Node	2	13	14	ReportYr3	0,5	46	243.674
3	7	Node	3	15	16	FinTime2	0,5	4340	19.989
3	8	Node	3	17	18	ReportTime	0,5	692	54.986
3	9	Node	4	19	20	InjScoreTot	87,5	1226	54.457
3	10	Node	4	21	22	InjScoreTot	62,5	162	112.333
3	11	Node	5	23	24	FinTime2	0,5	101	78.539
3	12	Node	5	25	26	ReportTime	0,5	53	144.538
3	13	Node	6	27	28	ReportYr2	0,5	43	266.160
3	14	Leaf	6	-	-	-	-	3	83.462
4	15	Node	7	29	30	ReportTime	0,5	2898	13.545
4	16	Node	7	31	32	ReportTime	0,5	1442	33.727
4	17	Node	8	33	34	InjScoreTot	62,5	523	49.081
4	18	Node	8	35	36	ReportYr2	0,5	169	72.337
4	19	Node	9	37	38	FinTime2	0,5	1097	49.800
4	20	Node	9	39	40	ReportYr3	0,5	129	100.949
4	21	Node	10	41	42	InjScoreTot	37,5	116	88.581
4	22	Node	10	43	44	InjNb2	0,5	46	158.795
4	23	Leaf	11	-	-	-	-	38	46.638
4	24	Node	11	45	46	LegalBin	0,5	63	89.273
4	25	Node	12	47	48	LegalBin	0,5	43	133.284
4	26	Leaf	12	-	-	-	-	10	195.888
4	27	Node	13	49	50	ReportYr1	0,5	39	286.621
4	28	Leaf	13	-	-	-	-	4	92.235
5	29	Node	15	51	52	FinTime1	0,5	1964	11.129
5	30	Node	15	53	54	FinTime1	0,5	934	18.529
5	31	Node	16	55	56	InjScoreTot	87,5	1119	29.225
5	32	Node	16	57	58	InjScoreTot	62,5	323	48.668
5	33	Node	17	59	60	ReportYr2	0,5	248	39.801
5	34	Node	17	61	62	ReportYr3	0,5	275	59.829
5	35	Node	18	63	64	ReportYr3	0,5	118	81.247
5	36	Node	18	65	66	InjScoreTot	62,5	51	50.768
5	37	Node	19	67	68	ReportTime	2,5	825	42.349
5	38	Node	19	69	70	ReportYr3	0,5	272	72.449
5	39	Node	20	71	72	ReportTime	4,5	108	114.072
5	40	Leaf	20	-	-	-	-	21	33.458
5	41	Node	21	73	74	ReportTime	2,5	76	82.659
5	42	Node	21	75	76	ReportTime	4,5	40	102.930
5	43	Node	22	77	78	InjNb5	0,5	46	141.836
5	44	Leaf	22	-	-	-	-	0	282.132
5	45	Leaf	24	-	-	-	-	14	124.178
5	46	Node	24	79	80	ReportYr1	0,5	49	80.238
5	47	Leaf	25	-	-	-	-	12	171.149
5	48	Leaf	25	-	-	-	-	31	114.737
5	49	Leaf	27	-	-	-	-	27	350.655
5	50	Leaf	27	-	-	-	-	12	208.666

Fig. 3.29: Best pruned regression tree features until the fifth level

teen predictors. As discussed in Subsection 3.4.4, we may also accept the assumption of one single hidden layer. However, how many hidden neurons shall we use? Intuitively, too few hidden neurons will not be able to detect information, while too many hidden neurons will imply overfitting. Just like for the regression tree, let's try various cases by gradually increasing the number of hidden neurons, say from 1 to 20. The results are summarized in Figure 3.30 (the fields have the same meaning as in Figure 3.28). Once again, training error and validation error are very close: the neural network is not overfitting. The bad performance due to few hidden neurons - be-

number of hidden neurons	average training error	average validation error	average overall error	percentage weighted error	percentage overall error
1	30.420	30.068	30.279	16,23%	12,50%
2	29.634	29.621	29.629	15,20%	9,17%
3	29.706	29.838	29.758	13,96%	12,09%
4	28.858	28.947	28.894	8,96%	8,57%
5	29.028	29.285	29.131	11,26%	9,27%
6	28.734	28.828	28.771	10,08%	8,52%
7	28.848	29.338	29.044	11,13%	9,80%
8	**28.068**	**28.607**	**28.284**	**8,30%**	**5,57%**
9	28.609	29.273	28.875	8,89%	8,28%
10	28.615	29.411	28.933	10,66%	6,89%
11	28.318	29.056	28.613	10,04%	8,81%
12	28.553	29.229	28.824	10,60%	8,02%
13	28.376	28.975	28.616	9,81%	8,33%
14	28.375	29.368	28.772	9,82%	7,10%
15	28.261	28.900	28.516	8,82%	6,44%
16	28.480	29.020	28.696	9,64%	7,19%
17	28.402	29.299	28.761	10,07%	7,77%
18	28.188	28.939	28.488	9,00%	6,90%
19	28.321	28.831	28.525	8,64%	6,55%
20	28.044	28.890	28.382	9,06%	5,76%

Fig. 3.30: Neural network performance varying by number of hidden neurons

tween 1 and 7 - is quite clear. By adding more hidden neurons, the error tends to decrease slowly, but it seems that it is not overfitting by 20 neurons yet: it will probably start overfitting a bit later. The chosen neural network is that (in bold in Figure 3.28) leading to the lowest percentage weighted error, which also coincides to the lowest average overall error and the lowest percentage overall error. A formal representation of this neural network is in Figure 3.31, while the related parameters are reported in Figure 3.32.
As a conclusion to this subsection, we will present some global results. First of all, the convergence plots of the regression tree and the neural network in Figure 3.33 and 3.34 respectively. More importantly, we compare actual data and estimations in Figure 3.35-3.38. Remember that the chosen regression tree and neural network got a weighted error of 3,67% and 8,30% respectively, which relate to the lowest validation errors in Figure 3.30 and 3.28, so their good punctual estimations for the claim payments are not surprising. Let's analyse them by closing delay.
For closing delay 0, GLM and regression tree seem more accurate than neural network, as we can observe by comparing the estimated amounts of reporting year 1994, for instance. To some extent, this is the case for closing delay 1 as well, whatever the reporting year. For both those closing delays, re-

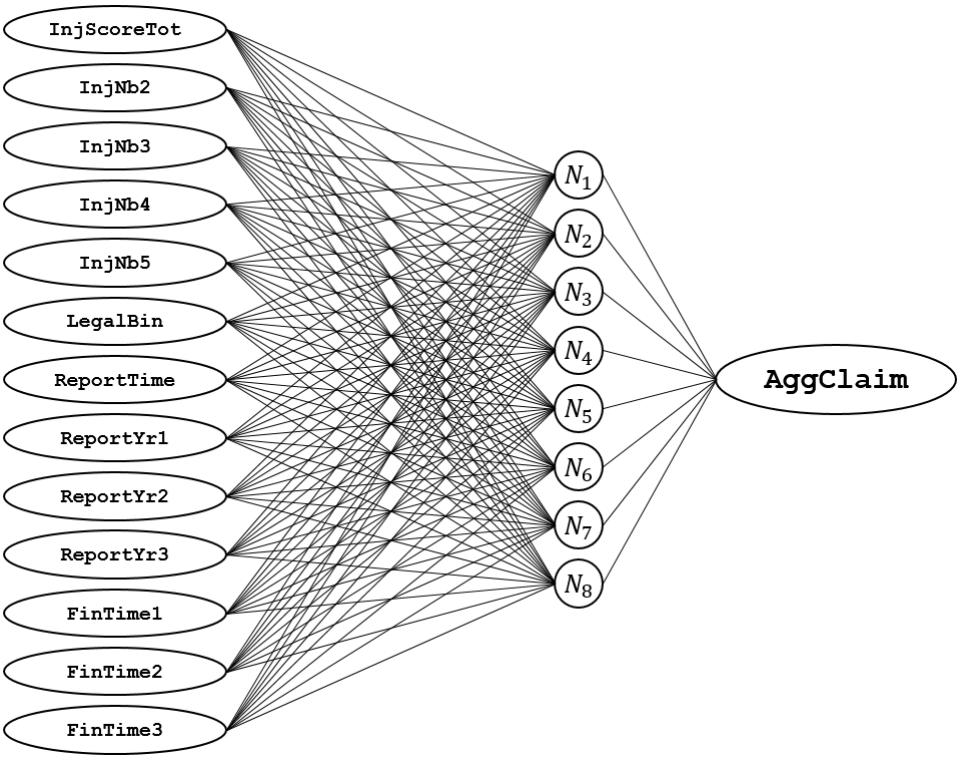


Fig. 3.31: 8-hidden-neuron neural network for claim amount prediction

	Bias	1,6433	0,9728	-2,6430	3,4999	-0,4701	-0,8636	0,6062	1,2669
	InjScoreTot	-3,6164	1,6972	-1,7925	-7,9758	-1,0843	3,7490	-2,8758	-1,2362
	InjNb2	2,5818	1,4252	2,8352	-1,1509	-0,5093	-0,7649	-0,7377	-0,3781
	InjNb3	-1,7530	-2,0918	1,9940	-2,4436	1,1031	0,3761	1,7829	0,2160
	InjNb4	3,7145	1,6844	-0,2911	1,8491	-0,1809	-1,6364	0,5876	0,3556
	InjNb5	1,5213	-0,3456	2,8264	-1,0558	0,0054	-1,2920	1,6087	-1,5291
	LegalBin	-1,4256	-1,0866	0,2715	-0,2891	-1,0853	-0,0509	0,5342	-2,8530
	ReportTime	-6,4297	-3,9165	-0,2582	0,6026	-0,6748	-2,5407	-1,1021	-2,6102
	ReportYr1	4,7466	3,8977	0,5149	-2,6189	-0,4819	0,2611	2,5912	1,5195
	ReportYr2	-2,8519	-2,7222	0,8436	-1,0376	-0,3715	0,0108	1,1027	-1,6984
	ReportYr3	-1,7670	-2,6942	-0,4083	-2,1818	-1,1539	-0,0667	1,8782	0,1445
	FinTime1	1,0403	1,3477	-0,3577	-0,7600	0,1805	0,1251	0,8446	0,2697
	FinTime2	1,5695	1,8607	-0,6853	-0,7159	0,5422	-0,6917	1,0201	0,9820
	FinTime3	-1,4015	-0,6039	0,5690	-0,4504	-0,3393	-2,9462	-2,0530	1,1273
	Bias	*Neuron 1*	*Neuron 2*	*Neuron 3*	*Neuron 4*	*Neuron 5*	*Neuron 6*	*Neuron 7*	*Neuron 8*
Response	0,3426	-2,8543	4,0754	-1,9584	-2,4236	1,7553	-3,2518	-2,0831	-1,4936

Fig. 3.32: 8-hidden-neuron neural network parameters

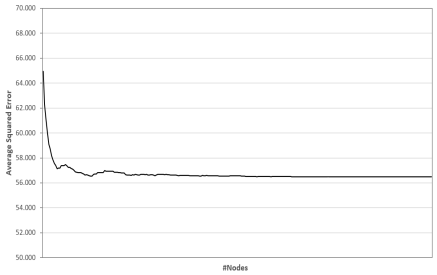


Fig. 3.33: Convergence of the error for the regression tree

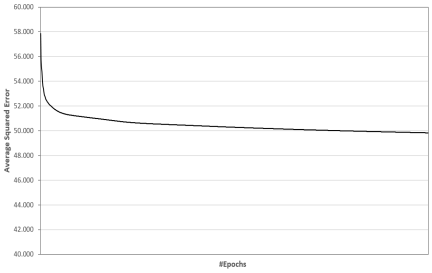


Fig. 3.34: Convergence of the error for the neural network

reporting year	closing delay 0	1	2	3
1993	26.038.265	77.236.474	74.285.051	58.268.484
1994	16.746.725	42.454.730	42.485.823	55.798.479
1995	6.076.308	20.958.156	41.895.020	50.580.334
1996	4.090.537	21.665.535	46.736.025	8.158.885

Fig. 3.35: Actual data

reporting year	closing delay 0	1	2	3
1993	23.942.277	72.096.426	85.372.988	68.135.627
1994	15.615.349	43.607.867	49.123.608	69.977.076
1995	6.294.610	21.590.823	41.199.048	51.380.998
1996	4.338.858	23.040.826	46.178.322	10.158.513

Fig. 3.36: Gamma regression

reporting year	closing delay 0	1	2	3
1993	27.777.729	74.715.426	74.860.543	53.671.653
1994	17.317.427	44.645.051	41.115.866	56.014.481
1995	7.819.221	22.443.511	40.888.916	51.092.375
1996	5.285.040	22.293.701	46.183.852	9.048.303

Fig. 3.37: Regression tree

reporting year	closing delay 0	1	2	3
1993	30.323.454	80.122.598	74.090.121	56.198.161
1994	21.876.621	49.791.426	44.957.040	51.693.100
1995	8.007.124	25.605.931	47.106.288	49.418.199
1996	5.358.636	27.349.566	46.177.586	8.455.383

Fig. 3.38: Neural network

gression tree and neural network tend to overestimate the claim payments, although this is much more evident for the latter.
For closing delay 2, we can highlight the good performance of the regression tree. By contrast, GLM and neural network return good estimations for some reporting years only: the former for 1995 and 1996, while the latter for 1993 and 1996. Finally, the estimations for closing delay 3 report a major performance gap between the GLM and the two other methods, as it is clear from the amounts for all the reporting years except for 1995.
Densities and QQ-plots in Figures 3.39-3.44 somewhat confirms that both regression tree and neural network outperform the GLM. Comparing Figures 3.40 and 3.42, for instance, it is worth noting how the accuracy of non-GLM methods tend to improve along the tail of the distribution, that is, at high closing delays. Given their nonparametric or semiparametric nature, they have more chance of adapting to any target variable, in spite of its peculiarities. On the other hand, this is not the case for the given GLM, although we might have tested other underlying distributions such as the inverse Gaussian or any Tweedie distribution (see Subsection 1.6.1).

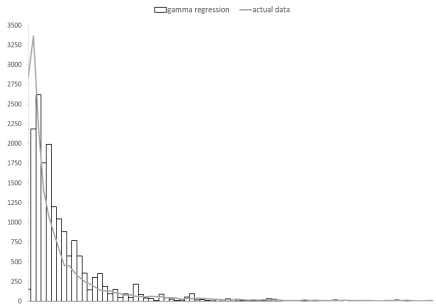


Fig. 3.39: Density
actual vs. gamma regression

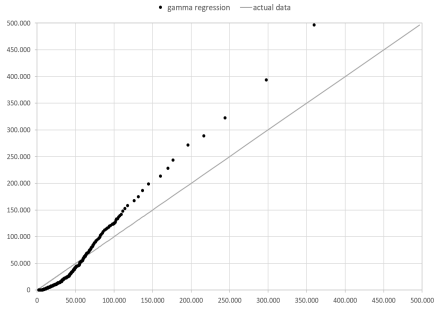


Fig. 3.40: QQ-plot
actual vs. gamma regression

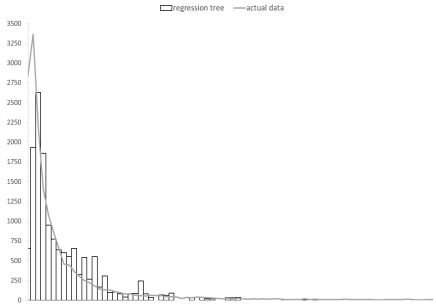


Fig. 3.41: Density
actual vs. regression tree

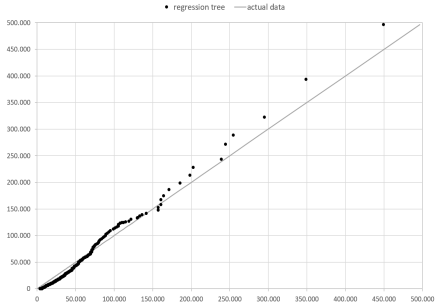


Fig. 3.42: QQ-plot
actual vs. regression tree

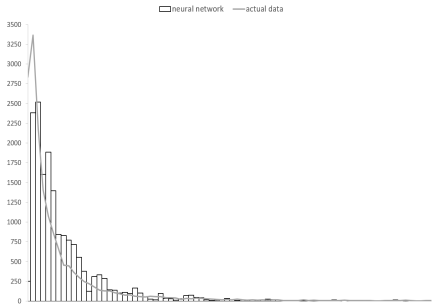


Fig. 3.43: Density
actual vs. neural network

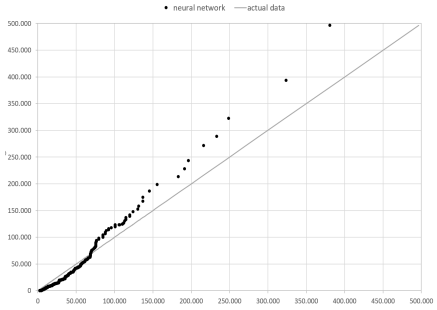


Fig. 3.44: QQ-plot
actual vs. neural network

3.5.4 Claim reserve estimation as an ensemble

In data science, an *ensemble* is a complex machine learning algorithm consisting of a number of simpler machine learning tools. The combination may be very easy to implement (for instance, relating to the three predictive models in Subsection 3.5.3, an ensemble for the prediction of the claim amount could be the average of the three different predictions), or very difficult. In any case, the goal is the improvement of the predictive performance.
Performance assessment for an ensemble typically requires more computation than performance assessment for its constituents, so ensembles may be thought of as a way to compensate for poor learning algorithms by performing a lot of extra computation. Therefore, fast algorithms such as CARTs are commonly used in some ensemble versions (e.g., *bagging trees*, *random forests* and *boosting trees*, introduced in the next chapter) which could have been used to estimate reserve amount. However, we decided to avoid it because this chapter mostly focuses on fundamental machine learning tools.
From this perspective, the estimation expressed by (3.4) may be seen exactly this way, that is, as an ensemble resulting from the combination of a classification tool and a regression tool. Of course, considering all the tools described in Section 3.4, any of the classification methods may be combined with any of the regression methods. The choice will depend on the performance reported in Figure 3.45-3.48 by reporting year, and the overall performance reported in Figure 3.49.
Globally, gamma regression significantly overestimates the claim payments, especially for the reporting year 1993 and 1994. More importantly, the major problem is that the performance is quite different across reporting years regardless of the classification tool used. All in all, the best performances seem to be related to two ensembles:

- multinomial regression and regression tree (overall error 0,19%)
- classification tree and regression tree (overall error -0,14%)

and their error per reporting year is always lower than 3% in absolute value. Basically, if one of these two combinations is used to predict the payment as soon as the claim is reported (assuming that all the relevant information is immediately available), we will reserve a very accurate amount for any reporting year at an aggregated level. That sounds quite good, but it is not necessarily the best solution as a proper reserve should always account for some level of conservatism, as long as it is uniformly included into each allocation. For instance, Figure 3.45-3.48 shows that the two ensembles

- multinomial regression and neural network (overall error 6,36%)
- classification tree and neural network (overall error 5,76%)

Actual claim amount 235.828.275	*gamma regression*	*regression tree*	*neural network*
multinomial regression	252.028.878	230.418.987	241.288.489
naive Bayes	239.075.514	217.568.447	231.965.733
nearest neighbours	249.685.356	229.945.901	238.026.479
classification tree	258.083.300	233.422.075	243.846.286

Δ%	*gamma regression*	*regression tree*	*neural network*
multinomial regression	6,87%	-2,29%	2,32%
naive Bayes	1,38%	-7,74%	-1,64%
nearest neighbours	5,88%	-2,49%	0,93%
classification tree	9,44%	-1,02%	3,40%

Fig. 3.45: Predictive performance for reporting year 1993

Actual claim amount 157.485.757	*gamma regression*	*regression tree*	*neural network*
multinomial regression	186.011.183	159.763.704	172.850.776
naive Bayes	177.763.171	149.619.919	165.548.345
nearest neighbours	182.202.506	159.124.199	169.743.174
classification tree	182.470.513	156.827.235	170.113.669

Δ%	*gamma regression*	*regression tree*	*neural network*
multinomial regression	18,11%	1,45%	9,76%
naive Bayes	12,88%	-4,99%	5,12%
nearest neighbours	15,69%	1,04%	7,78%
classification tree	15,86%	-0,42%	8,02%

Fig. 3.46: Predictive performance for reporting year 1994

Actual claim amount 119.509.819	*gamma regression*	*regression tree*	*neural network*
multinomial regression	118.524.978	122.550.580	129.893.618
naive Bayes	126.742.857	129.321.735	136.983.874
nearest neighbours	117.758.364	120.167.319	128.321.910
classification tree	116.673.918	119.593.669	126.714.629

Δ%	*gamma regression*	*regression tree*	*neural network*
multinomial regression	-0,82%	2,54%	8,69%
naive Bayes	6,05%	8,21%	14,62%
nearest neighbours	-1,47%	0,55%	7,37%
classification tree	-2,37%	0,07%	6,03%

Fig. 3.47: Predictive performance for reporting year 1995

Actual claim amount 80.650.982	*gamma regression*	*regression tree*	*neural network*
multinomial regression	82.330.316	81.844.353	87.189.365
naive Bayes	87.835.102	86.321.719	91.700.340
nearest neighbours	83.617.031	81.500.277	87.200.645
classification tree	83.901.388	82.788.649	87.006.662

Δ%	*gamma regression*	*regression tree*	*neural network*
multinomial regression	2,08%	1,48%	8,11%
naive Bayes	8,91%	7,03%	13,70%
nearest neighbours	3,68%	1,05%	8,12%
classification tree	4,03%	2,65%	7,88%

Fig. 3.48: Predictive performance for reporting year 1996

Actual claim amount 593.474.833	*gamma regression*	*regression tree*	*neural network*
multinomial regression	638.895.354	594.577.625	631.222.248
naive Bayes	631.416.644	582.831.820	626.198.292
nearest neighbours	633.263.257	590.737.696	623.292.207
classification tree	641.129.119	592.631.628	627.681.248

Δ%	*gamma regression*	*regression tree*	*neural network*
multinomial regression	7,65%	0,19%	6,36%
naive Bayes	6,39%	-1,79%	5,51%
nearest neighbours	6,70%	-0,46%	5,02%
classification tree	8,03%	-0,14%	5,76%

Fig. 3.49: Overall predictive performance

		Reporting year 1997 (closing delay 0-2 years)	Reporting year 1998 (closing delay 0-1 years)
	Actual claim amount	**46.006.511**	**9.683.722**
multinomial regression & regression tree	Predicted amount	63.673.036	22.975.836
	Delta	17.666.525	13.292.114
classification tree & regression tree	Predicted amount	64.860.982	24.113.611
	Delta	18.854.471	14.429.888
multinomial regression & neural network	Predicted amount	68.247.260	23.092.487
	Delta	22.240.749	13.408.764
classification tree & neural network	Predicted amount	68.079.346	23.632.134
	Delta	22.072.834	13.948.412

Fig. 3.50: Summary results on the test dataset (reporting years 1997-1998)

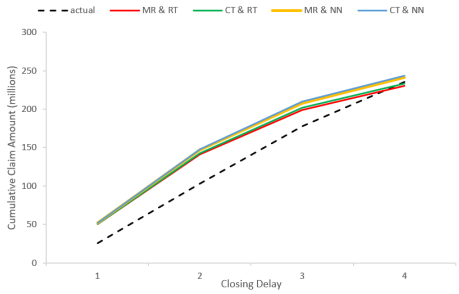


Fig. 3.51: Cumulative amount for claims reported in 1993

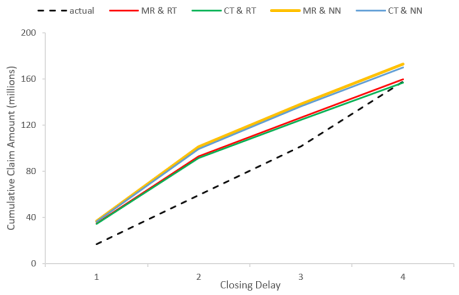


Fig. 3.52: Cumulative amount for claims reported in 1994

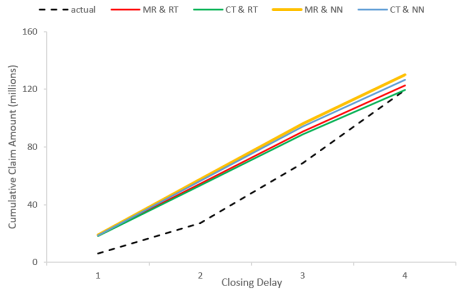


Fig. 3.53: Cumulative amount for claims reported in 1995

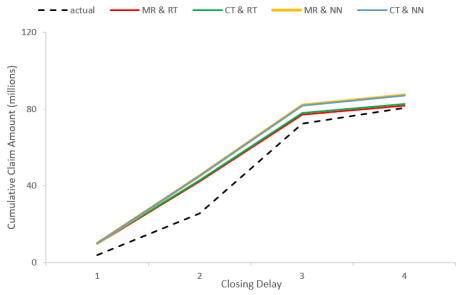


Fig. 3.54: Cumulative amount for claims reported in 1996

slightly overestimate the claim payments, but still less than gamma regression. In particular, the overestimation is quite stable over the reporting years 1994-1996, that is, they overestimate in the same direction, with similar magnitude, regardless of the reporting year. If the company reserves using one of the two aforementioned ensembles, it will most probably overestimate the amount by around 6-8% (we do not consider the lower but older - thus less significant - errors for the year 1993). For our purpose, it could really be the best compromise, including a reasonable prudence margin.
Once the best ensembles are selected, the last step is the estimation of the reserve on the test dataset, that is, the allocation for the claims reported in 1997 and 1998. A further assumption is immediately necessary: given that the reporting year is a categorical variable, the new reporting years are not included in our models, so they will be replaced by the last year available, that is, 1996 - the most significant one in terms of timing. The results are stored in Figure 3.50. As specified in the column fields, remember that these two reporting years contain information for a limited number of closing delay categories: 0, 1 and 2 for 1997, while 0 and 1 for 1998. This is because the dataset was extracted at year end 1999. Whatever the ensemble is, the overall claim amounts are materially overestimated for both of the years. A total 46M claim amount in 1997 is predicted to be about 50% greater, between 63M and 68M. In 1998, this delta reaches about 150%! Actually, this is not surprising, if we take into account the performance of both multinomial regression and classification tree in predicting closing delay (see Subsection 3.5.2). The ensembles correctly predict amounts, but they tend to allocate them in wrong closing delay categories. Indeed, some claims are allocated as lower delays - 0 and 1 - whereas they should be allocated as higher delays - 2 and 3. Additionally, we do not have many actual claims by higher delay categories yet, so this bias is not balanced out by them, as opposed to the reporting years 1993-1996. However, this happens in those years as well, if we separate claims by reporting year. In Figure 3.51-3.54), this effect is represented by the gap between the dashed lines (actual cumulative amount paid) and the four coloured lines, representing the four ensembles.
On the one hand, the low-closing-delay payments tend to be overestimated, but the overall amount per reporting year still converges to the correct one. As it is said, keeping in mind that a prudence margin is always required, such ensembles may be still used for reserving purposes. On the other hand, it would be legitimate to let the company reduce such a prudence margin. Ideally, it should own enough data to get better predictions for the closing delay, in order to reduce the gaps shown in Figure 3.51-3.54.

3.5.5 Small claims and large claims

A common enhancement in non-life modelling is based on the initial distinction between small claims and large claims. It allows for an *ad hoc* treatment

Actual Class	Predicted Class 0	1	Cases Number	Errors Number	Errors %
0	8.121	1.395	9.516	1.395	14,66%
1	116	299	415	116	27,95%
Total			**9.931**	**1.511**	**15,21%**

Fig. 3.55: Small claim results for LR (training dataset)

Actual Class	Predicted Class 0	1	Cases Number	Errors Number	Errors %
0	5.463	912	6.375	912	14,31%
1	81	164	245	81	33,06%
Total			**6.620**	**993**	**15,00%**

Fig. 3.56: Small claim results for LR (validation dataset)

Actual Class	Predicted Class 0	1	Cases Number	Errors Number	Errors %
0	8.074	1.442	9.516	1.442	15,15%
1	134	281	415	134	32,29%
Total			**9.931**	**1.576**	**15,87%**

Fig. 3.57: Small claim results for NB (training dataset)

Actual Class	Predicted Class 0	1	Cases Number	Errors Number	Errors %
0	5.405	970	6.375	970	15,22%
1	80	165	245	80	32,65%
Total			**6.620**	**1.050**	**15,86%**

Fig. 3.58: Small claim results for NB (validation dataset)

Actual Class	Predicted Class 0	1	Cases Number	Errors Number	Errors %
0	8.104	1.412	9.516	1.412	14,84%
1	155	260	415	155	37,35%
Total			**9.931**	**1.567**	**15,78%**

Fig. 3.59: Small claim results for NN (training dataset)

Actual Class	Predicted Class 0	1	Cases Number	Errors Number	Errors %
0	5.443	932	6.375	932	14,62%
1	85	160	245	85	34,69%
Total			**6.620**	**1.017**	**15,36%**

Fig. 3.60: Small claim results for NN (validation dataset)

that, on one hand, improves accuracy, and, on the other hand, avoids undesired reserve estimations. For example, the claims department might decide to assign a predefined reserve amount for all the claims that will be likely smaller than some amount. On the other hand, the claims and actuarial departments might use appropriate *ad hoc* methods to estimate reserve for all the claims that will be likely larger than some amount.

In this subsection, we will perform the separation on our dataset by defining two further binary variables, `SmallClaimBin` and `LargeClaimBin`, the targets for our classification. The former equals 1 one if the claim amount is lower than 1.000, while the latter equals 1 if the claim amount is greater than 100.000. Notice that a single, categorical variable with three categories - small, normal and large claims - would be much more efficient, in the sense that we might run one single algorithm to classify a claim in one of them. However, we prefer the separation of the two problems since binary targets allow for an easier visualization of the results (e.g., ROC curves).

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	6.619	2.535	9.154	2.535	27,69%
1	202	575	777	202	26,00%
Total			**9.931**	**2.737**	**27,56%**

Fig. 3.61: Large claim results for LR (training dataset)

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	4.439	1.664	6.103	1.664	27,27%
1	156	361	517	156	30,17%
Total			**6.620**	**1.820**	**27,49%**

Fig. 3.62: Large claim results for LR (validation dataset)

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	6.668	2.486	9.154	2.486	27,16%
1	208	569	777	208	26,77%
Total			**9.931**	**2.694**	**27,13%**

Fig. 3.63: Large claim results for NB (training dataset)

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	4.474	1.629	6.103	1.629	26,69%
1	162	355	517	162	31,33%
Total			**6.620**	**1.791**	**27,05%**

Fig. 3.64: Large claim results for NB (validation dataset)

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	6.649	2.505	9.154	2.505	27,37%
1	191	586	777	191	24,58%
Total			**9.931**	**2.696**	**27,15%**

Fig. 3.65: Large claim results for NN (training dataset)

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	4.423	1.680	6.103	1.680	27,53%
1	153	364	517	153	29,59%
Total			**6.620**	**1.833**	**27,69%**

Fig. 3.66: Large claim results for NN (validation dataset)

We will use some of the tools which have already been used in the previous sections, properly adapted for classification, that is, logistic regressions, naïve Bayes and neural networks. For sake of consistency, the logistic regressions will include all the predictors selected by the stepwise algorithm in Figure 3.27, and the neural networks will maintain the same architecture as in Figure 3.31.
For the small claim analysis, the cut-off values have been set in order to approximately return the same complement of the specificity in the training dataset among the three tools, that is, the same number of false positive. As you can see in Figure 3.55, 3.57 and 3.59, that is always around 15%. In this way, we can control the misclassification error on the majority of the records - non-small claims - and then observe it in the rest of the dataset - small claims - which represents the most interesting part. In fact, misclassification errors on the validation dataset in Figure 3.56, 3.58 and 3.60 do not demonstrate any relevant difference. The sensitivity reveals that about two

small claims out of three small claims are successfully detected, while the false positive percentage and the overall error are still around 15%. These are quite good results: the ROC curves in Figure 3.67-3.68 imply AUCs greater than 85%, and the decile charts in Figure 3.69-3.70 reveal predictive powers that are at least four times higher than that of the naïve rule in the first decile.

For the large claim analysis, the cut-off values have also been set in order to approximately return the same complement of the specificity in the training dataset among the three tools, that is, the same number of false positive. As you can see in Figure 3.61, 3.63 and 3.65, that is always between 27% and 28%. In this way, we can control the misclassification error on the majority of the records - non-large claims - and then observe it in the rest of the dataset - large claims - which represents the most interesting part. Similarly to the small claim analysis, misclassification errors on the validation dataset in Figure 3.62, 3.64 and 3.66 do not demonstrate any relevant difference. The sensitivity reveals that about two small claims out of three small claims are successfully detected, while the false positive percentage and the overall error are still around 27%. These are quite good results too, although slightly worse than those from the small claim analysis: the ROC curves in Figure 3.67-3.68 imply AUCs greater than 77%, and the decile charts in Figure 3.69-3.70 reveal predictive powers that are around four times higher than that of the naïve rule in the first decile.

All in all, we reach a very good accuracy with all the tools we used. Nonetheless, it is worth noting that it does not depend on the complexity of the underlying algorithm. In other words, there is no reason to prefer logistic regression or neural network if a simple naïve Bayes returns comparable accuracy. Logistic regressions and neural networks always require preliminary analysis such as variable selections, optimizations and calibrations since they depend on parameters and meta-parameters. By contrast, naïve Bayes is purely non-parametric, and there is really no algorithm behind that. If it performs as well as more complex tools, why should we use the latter? This is the reason why naïve Bayes is still used for many practice applications (e.g., spam filtering) in industry.

A claim department may hypothetically use our results to differentiate claim treatment. Moreover, actuaries may separately build reserving algorithms for small claims and large claims, for example, using the techniques presented in this chapter. We did not proceed this way for two reasons essentially. First, we cannot rely on very big data, that is, 660 small claims and 1294 large claims. Second, it would not change the idea behind our analysis, that is, improving reserving through machine learning. Moreover, this separation could be even redundant when it comes with tools that are already characterized by discrimination features such as CARTs. In other words, it could be even embedded in the algorithm itself, whether implicitly or explicitly. On the other hand, the separation may be crucial when

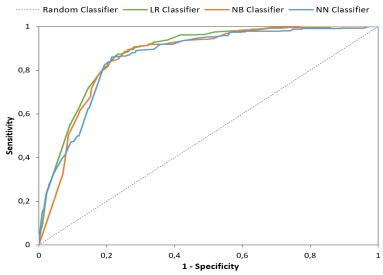


Fig. 3.67: Small claim ROCs (training dataset)

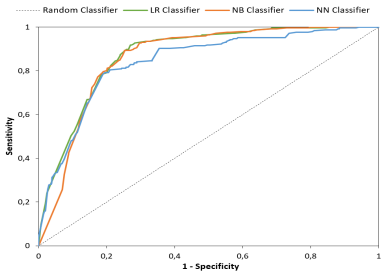


Fig. 3.68: Small claim ROCs (validation dataset)

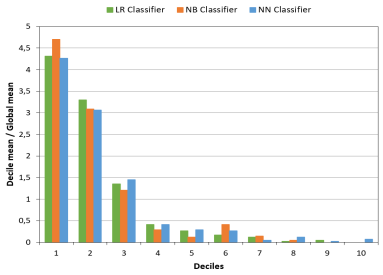


Fig. 3.69: Small claim deciles (training dataset)

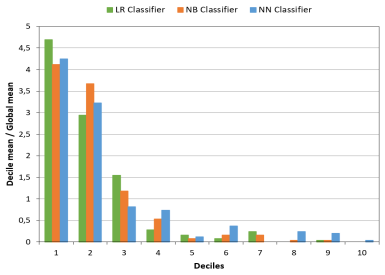


Fig. 3.70: Small claim deciles (validation dataset)

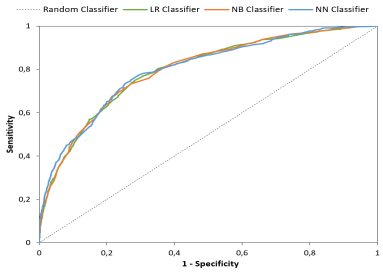


Fig. 3.71: Large claim ROCs (training dataset)

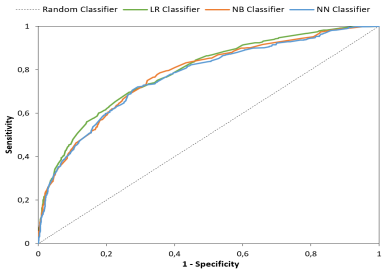


Fig. 3.72: Large claim ROCs (validation dataset)

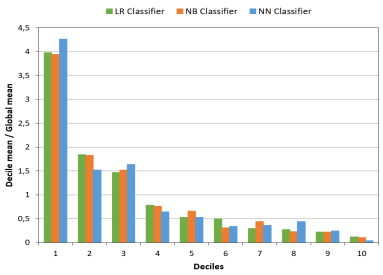


Fig. 3.73: Large claim deciles (training dataset)

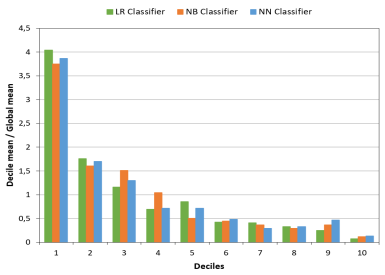


Fig. 3.74: Large claim deciles (validation dataset)

we use tools that are sensitive to outliers, especially regressions. Indeed, it is a common practice among non-life actuaries, who always use regression models.

3.6 Limitations, extensions and conclusions

In this chapter, we exploited the potential of machine learning in solving a traditional problem such as case reserving for a specific type of claims in non-life. According to the CRISP-DM standard for the data mining process (see Section 1.2), the chapter can be formally broken as follows:

C1 *Business understanding*: Section 3.1 and 3.2

C2 *Data understanding*: Subsection 3.5.1

C3 *Data preparation*: Subsection 3.5.1

C4 *Modelling*: Subsection 3.5.2 and 3.5.3

C5 *Evaluation*: Subsection 3.5.4 and 3.5.5

C6 *Deployment*: none.

Given that we focused on a specific application to automobile bodily injury claim data, no general conclusion may be drawn (which is typical when it comes with machine learning), but of course the ensembles we used can be easily adapted to different datasets. In addition, we chose two representative from the class of regression models - multinomial regression and gamma regression - but there may be other, more complex regression models leading to better results (e.g., GLMs based on Tweedie distributions). Nonetheless, the *a priori* choice of the underlying probabilistic model (e.g., link functions, distributions, etc.) is a typical issue of regression. Such an issue is completely skipped in machine learning, somewhat replaced by the proper setting of few meta-parameters (e.g., minimum records in the leaves of a tree, number of hidden neurons, etc.). This is feasible thanks to the unique flexibility of machine learning tools.
On the other hand, we should point out some major drawbacks we faced during the analysis:

- as discussed in Section 3.1, case reserving is just one instance of the reserve allocation in non-life, and applying machine learning to other instances (e.g., aggregated estimation methods such as run-off triangles) might be much less immediate, or not practicable at all;
- data availability is the crucial constraint, and it may happen that a dataset is extremely useful for some target variables, but very poor for others (for instance, compare closing delay performance in Subsection 3.5.2 and payment amount performance in Subsection 3.5.3);

- poor predictions for the closing delay is not a problem if claim reserves are evaluated as at REPORTING year, but it would if they were evaluated as at PAYMENT year (this is actually another way to state the problem we faced in Subsection 3.5.4);
- IBNYR reserve might be a relevant component of the total claim reserve, but it is not considered here (see Section 3.2 for further details).

Data is always a major issue in machine learning. In our application, we experienced it when selecting useful data in Section 3.5.1. A number of records (in grey in Figure 3.8, 3.9 and 3.10) have been excluded, for instance, those related to closing delays beyond three years. Even if they are few, actuaries know they bring relevant information about long-lasting claims. Generally speaking, when records are few, pure machine learning techniques should be avoided in favour of less data-driven methods such as regressions and extrapolations. They may actually help us to deal with the variability explained by those records. For instance, using Poisson regression instead of multinomial regression, we can allow for closing delays that do not appear in the dataset.
Regarding the last bullet point, we should probably build a completely different model for IBNYR prediction. However, it would not be strictly "individual", because the company does not get any individual claim data before the reporting date of the claim itself. If we want to use machine learning techniques for this purpose, we should rather rely on different data. More specifically, cross-sectional data related to the policyholder (age, family, address, habits, etc.) and external information like economic environment (unemployment rate, inflation rate, financial distress, etc.), weather conditions, natural catastrophes (storm, flood, earthquake, etc.) and so on. Of course, such additional data may be useful for better prediction of the RBNYS reserve as well.
It is our opinion that a complete analysis should distinguish between small claims and large claims just like in Subsection 3.5.5. Although it may be redundant when the algorithm manages to separate them automatically, those claims are sometimes distinguished by unique features, which could lead to very different treatment in reserving. This is also the case for zero claims, that is, claims that are closed without any payment: if they can be detected such as small claims, they will require some specific reserving process too.
A final remark is important to conclude the chapter. Even if data is materially informative, traditional parametric methods could still outperform nonparametric tools, or return comparable results. For instance, look at the payment amount predictions for closing delays 0 and 1 in Figure 3.35-3.38. Actually, gamma regression predictions seem more accurate than those of regression tree and neural network. That is just an example of the fact that machine learning is not always superior to traditional methods. The strength of machine learning relates to the ability to catch intricate depen-

dencies among data, which is however not always necessary. More importantly, a greater effort in predicting such dependencies could paradoxically lead to worse performance on regular data. This is also clear in Figure 3.35-3.38: better predictions for rare, severe claims (closing delays 3 and 4), but slightly worse predictions for numerous, regular claims (closing delays 0 and 1). In fact, if data is actually regular enough to fulfil the regression assumptions about residuals and multicollinearity, there is no reason to use other tools: regression would return the best performance by definition.
In spite of it, it is worth noting how these new methods can be convenient for non-life companies. Although we only used basic machine learning tools and combined them together, we have still got very accurate predictions per reporting year, outperforming regression.

3.6.1 Key conclusions for actuarial practitioners

This chapter demonstrated the primary importance of data availability. For instance, we could not use a considerable amount of records because of the data-driven nature of some nonparametric algorithms. Actually, if we want to avoid any assumption on the distribution of the target variable, data is all we have. A compromise may be represented by the usage of the same nonparametric tools to estimate parameters of predefined distributions (e.g., assuming the closing delay to be Poisson distributed, we could have estimated the related λ by using regression trees or neural networks), but it would make the model more rigid, so we did not deal with this option.
Data availability is as important as data significance: if information is not enough, performance will not be good regardless of the statistical model used. As we observed for closing delay estimation, all the methods we used returned approximately the same, significant error. Machine learning exploits information, but does not create it.
Nonetheless, if there is some relevant information in our dataset, machine learning techniques can outperform more traditional methods. This is what we observed in claim amount estimation: the regression tree returned a better QQ-plot as well as a more accurate overall reserve allocation as compared to gamma regression. Actuaries need to know that it is not always the case as it strongly depends on the specific dataset. However, they should acknowledge that there are alternatives to GLMs, and sometimes they can lead to better results.
Regardless of the model performance, individual reserving is becoming increasingly important thanks to real-time data availability. Run-off triangles are still widely used in many applications, but individual reserving offers the opportunity to enhance them (e.g., as a backtesting tool) or replace them (e.g., in heterogeneous branches). This evolution may result in two major outcomes: instantaneous, accurate reserve estimation, and a brand new field for non-life reserving actuaries.

Chapter 4

Policyholder Behaviour Modelling using *Ensembles*

Even if data science tends to be more useful in non-life actuarial practice due to the nature of the related information and techniques, life practice can benefit from it as well. One of the possible fields of application is policyholder behaviour, that is, the individual reaction to relevant dynamics such as ageing and economic conditions.

Actuarial context

In life insurance, policyholder behaviour is reflected by the exercise of options embedded in a product. As demonstrated by several studies, the exercise likelihood can be affected by various features, related to policyholder profile, terms of tariff and market interest rates. Without a doubt, the impact on profitability and solvency may be material, especially under unfavourable scenarios.
There is a wide variety of embedded options in the insurance market. Some of the most common include guaranteed annuity, fund switching, paid-up and so on. However, the largest impact is due to the surrender option as it may be embedded in any type of product. It represents the main non-financial risk driver in life insurance, allowing the policyholder for lapse at any time before the maturity with the withdrawal of the actual reserve.

Chapter overview

Aware of the superior performance of decision trees in the last chapter, we will try to use them for lapse rate estimation with a dataset from the Italian insurance market. However, we will face a typical drawback of most of the basic machine learning techniques: instability. Indeed, overfitting is a major issue in data science. This is the reason why tree-based ensembles are often preferred over single trees, and we will use their most common forms (i.e.,

bagging, random forest and boosting) for our last application.
The extraction of relevant information from policyholder profile, tariff features and macroeconomic conditions is just one aspect of the case study. The second step involves the usage of estimated lapse rates to assess the actual impact on the profitability of a life product. More specifically, we will build an asset-liability management model based on a stochastic interest rate model. Given that the estimated lapse rates will vary by year-dependent variables such as sum assured and forward rate (among others), we will get different lapse rates per year and simulation. In other words, the policyholder will dynamically react to changes in features, year after year.
Using Monte Carlo simulations, we will estimate the profit of the product with and without dynamic lapse rates. Given that the surrender is essentially an embedded put option, the former case will result in a decrease in profitability. Such a decrease represents a marginal measure of the surrender option value.

4.1 Introduction

Following the introduction of the most recent regulatory frameworks, the traditional actuarial methods are being replaced, little by little, by more complex and structured models to estimate the economic value (in his several forms) of insurance companies and their business. So far, in life insurance, most of the effort focused on financial risks, which encompass the most relevant risk factors (e.g., interest rate risk, credit risk, liquidity risk, reinvestment risk, etc.) for life insurers. However, those frameworks themselves highlight that non-financial risks should be evaluated and monitored as well, including the risks due to the policyholder behaviour (PHB). Typically, policyholders are given a number of embedded options within their insurance contracts, and predicting the exercise likelihood is crucial to forecast the portfolio profitability.
In general, PHB refers to the policyholder's tendency to exercise any of the options embedded in its insurance contract. They include surrender option, guaranteed annuity option, dynamic premium increase option, product switching option, fund switching option, paid-up option and so on. For several reasons, the surrender option leads to the greatest impact within the portfolio. Indeed, it is embedded in almost all the insurance products. To some extent, it guarantees the fairness of the contract by allowing the policyholder to receive back its reserve, net of some penalty. As opposed to other aforementioned options, the surrender option can be generally exercised at any time prior to the maturity, just like an American option, so that the insurer is exposed on a continuous basis.
In the Solvency II Directive, for instance, PHB was assumed as a relevant source of risk from the very first Quantitative Impact Study (QIS). The

paragraphs TP.2.129-TP.2.135 in Solvency II Technical Specification (2014) introduce the concept of PHB within the Best Estimate Liability (BEL) evaluation. For instance, TP.2.130 states that

> *Any assumptions made by insurance and reinsurance undertakings with respect to the likelihood that policyholders will exercise contractual options, including lapses and surrenders, should be realistic and based on current and credible information. The assumptions should take into account, either explicitly or implicitly, of the impact that future changes in financial and non-financial conditions may have on the exercise of those options.*

Paragraph TP.2.131 is even more specific:

> *Assumptions about the likelihood that policyholders will exercise contractual options should be based on analysis of past policyholder behaviour and a prospective assessment of expected policyholder behaviour. The analysis should take into account the following:*
>
> *(a) how beneficial the exercise of the options was and will be to the policyholders under past circumstances (whether the option is out of or barely in the money or is in the money),*
>
> *(b) the influence of past and future economic conditions,*
>
> *(c) the impact of past and future management actions,*
>
> *(d) any other circumstances that are likely to influence a decision whether to exercise the option.*

In the Solvency II framework, PHB is not only supposed to be object of the BEL, but also a risk factor to take into account within the lapse risk module of the Solvency Capital Requirement (SCR). Paragraph SCR.7.43 in Solvency II Technical Specification (2014) indeed provides the following definition:

> *Lapse risk is the risk of loss or adverse change in liabilities due to a change in the expected exercise rates of policyholder options. The relevant options are all legal or contractual policyholder rights to fully or partly terminate, surrender, decrease, restrict or suspend insurance cover or permit the insurance policy to lapse.*

More recently, the International Accounting Standards Board (IASB) has issued the new accounting standard, IFRS 17, for insurance contracts (see IFRS 17 Insurance Contracts (2017)). It is going to replace the previous standard, IFRS 4, starting from January 2021, in order to converge to common accounting rules within the insurance industry.

Although risk-based exercises such as those of Solvency II can be very different to the evaluations underlying accounting principles, PHB is a relevant feature of the former as well as the latter. Indeed, dealing with future cash-flow estimation, paragraph B62 of IFRS 17 Insurance Contracts (2017) points out the following:

> *Many insurance contracts have features that enable policyholders to take actions that change the amount, timing, nature or uncertainty of the amounts they will receive. Such features include renewal options, surrender options, conversion options and options to stop paying premiums while still receiving benefits under the contracts. The measurement of a group of insurance contracts shall reflect, on an expected value basis, the entity's current estimates of how the policyholders in the group will exercise the options available, and the risk adjustment for non-financial risk shall reflect the entity's current estimates of how the actual behaviour of the policyholders may differ from the expected behaviour. This requirement to determine the expected value applies regardless of the number of contracts in a group; for example it applies even if the group comprises a single contract. Thus, the measurement of a group of insurance contracts shall not assume a 100 per cent probability that policyholders will:*
>
> *(a) surrender their contracts, if there is some probability that some of the policyholders will not; or*
>
> *(b) continue their contracts, if there is some probability that some of the policyholders will not.*

Solvency II and IFRS 17 represent only two of the numerous regulatory frameworks that explicitly require PHB analysis nowadays. As such, it is the object of this chapter, which will embed dynamic PHB into the stochastic profit evaluation of an insurance product. We will focus on the possible risk of loss due to unanticipated PHB for full (i.e., not partial) surrender. The adjective "unanticipated" refers to the exclusion of any PHB assumption from both pricing and reserving, just like it is still common in traditional actuarial practice.
After a broad review of the past studies about PHB risk factors in Section 4.2, we will start our analysis. We can distinguish three steps:

1. data preparation (see Subsection 4.5.1)
2. lapse rate prediction (see Subsection 4.5.2)
3. profit and TVOG analysis (see Subsection 4.5.3 and 4.5.4).

So in short, we will first predict lapse rates in a dynamical way (that is, different lapse rates in different scenario simulations), and then use them

as a contingency in a typical profit test for a specific insurance contract. Notice that PHB occurs *after* economic scenario simulation, because the former depends on the latter. In other terms, PHB will be treated as a deterministic function - represented by the machine learning algorithm - of the scenario simulation, among others. All in all, we will not build a stochastic model for PHB, so that no further computational burden will be caused beyond that coming from the economic scenario simulation.

4.2 Drivers of the policyholder behaviour

Because surrender activity can be so damaging to a single company or to the life insurance industry if it occurs *en masse*, research on widespread surrender activity and its possible determinants is especially important. In the last century, several studies and papers have been published about the very sources of PHB. They referred to the financial and insurance market of various countries worldwide. Although our analysis does not aim to detect the fundamental causes of PHB, it is worth recalling the most relevant results about them, covering almost one century. This will provide us with a good foundation to start analysing a lapse rate database of an insurance company.

An extensive description of the lapse rate research's early stage can be found in Richardson et al. (1951), which focuses on North American markets. By earlier twenties, almost one hundred years ago, some results had already demonstrated correlation between lapse rate and economic conditions, curiously right before the great depression after 1929. Nonetheless, researchers were well aware that market variables could not completely explain the effective duration of any product. As a consequence, several studies started focusing on policyholder-related variables, finding out that lapse could be correlated to income, occupation, sex, age, family, premium frequency and amount, and others. Another relevant finding of those years regards the effects of global economic distress on lapse rates. Briefly, each policyholder seems to have a tolerance threshold depending on its risk-propensity. It can be largely irrational, but it is also related to the actual economic condition: in time of economic distress, it is likely that most of policyholders feel beyond their threshold, which leads them to close their contracts. In other terms, lapse rate cannot be represented as a regular function of a global index market.

Without a doubt, three of the most comprehensive studies in that period were Cannon (1948), LIAMA (1948) and LIAMA (1949). Among others, they suggest that an insurance company can limit lapse from contract's inception by selecting quality business, recognizable from some objective indications of good persistence, especially age at issue, premium frequency and plan. Surprisingly, the author of Cannon (1948) concludes that the agent's

ability in picking quality business can be even more crucial than the actual economic condition itself.
Later, in the 1960s and 1970s, a couple of interesting empirical contributions have been produced by the Institute and Faculty of Actuaries. Both Crombie et al. (1979) and Patrick et al. (1969) are based on Scottish data, and the first one is meant to update the second one. To some extent, we can say that Patrick et al. (1969) is based on data of 1960s, while Crombie et al. (1979) is based on data of 1970s (this is the reason why both of them have been published at the end of the respective decade). As the titles suggest, such studies are exploratory in nature, but consider some of the variables which, on the base of the aforementioned sources, can drive surrender decisions. Sex, age at entry, occupation, purpose of assurance, calendar year (as a representation of variable economic conditions), sum assured, premium payment term and frequency, distribution channel, policy duration - all of them have been included in the analysis, but only duration and age at entry showed a significant correlation with the lapse rate.
All in all, by the end of the 1970s, lapse rates were fairly steady, with increases occurring during recessions, and decreases occurring during expansions. However, in the late 1970s, some important aspects started changing. Markets began to experience the highest increase in interest rates and volatility ever, while new, more complex insurance products were introduced. The contemporary improvement in financial literacy among policyholders led many of them to surrender their policies for more rational reasons such as interest rate arbitrage, preference for pure financial products, awareness about their own risk-propensity and so on. Surrender was no longer the natural, though irrational, response to the need of money during time of distress (the so-called Emergency Fund Hypothesis, for instance, in Russell et al. (2013)). In some cases, it turned to be the result of a precise, financial-oriented decision of the policyholder (the so-called Interest Rate Hypothesis, for instance, in Russell et al. (2013)). Of course, it could only worsen the position of intermediaries. Insurers were forced to liquidate bonds to meet surrender requests at precisely the time when the values of bond portfolios were depressed by high interest rates.
The increased volatility in economic conditions and financial markets made the correlation between PHB and macroeconomic variables much more interesting than policy features. A number of studies published in the last thirty years focuses on the macroeconomic determinants of global lapse rates in various countries. Most of them succeed in proving the Emergency Fund Hypothesis, but not the Interest Rate Hypothesis. The authors of Dar et al. (1989) explored the relationships between variables like interest rates and unemployment rate with surrender activity in the UK endowment life insurance market from the period 1952-1985. Something similar has been analysed by the author of Outreville (1990) using US and Canadian data of whole life policies from the period 1955-1979. In this study, the unemploy-

ment rate has a significantly positive effect on lapse rate, while policyholder's income has a significantly negative effect, while no significant relationship with interest rates was found. Further, the author of Hoyt (1994) concludes that the unemployment rate is the most significant variable in predicting surrender activity for universal life policies in the US from the period 1982-1986. The authors of Chen et al. (2003) used a dataset provided by the American Council of Life Insurance (ACLI) from the period 1951-1998 to confirm the strong correlation of the surrender activity with the unemployment rate. Nonetheless, they find a strong impact of the interest rate as well. In other words, Chen et al. (2003) supports both the Emergency Fund Hypothesis and Interest Rate Hypothesis.
In more recent years, other researchers focused on some European and Asian countries, both for empirical study and regression-based prediction of lapse rates. The Italian insurance market of savings products has been analysed in Cerchiara et al. (2009) by using surrender experience data of a large Italian bankassurer from the period 1991-2007. Explanatory variables included product type, calendar year, duration and inception year. In Kim (2010), the author used logistic regression to model lapse rate of Korean interest indexed annuities. Explanatory variables included the difference between reference market rates and product crediting rates, policy duration, unemployment rate, economy growth rate and some seasonal effects. One of the most comprehensive surrender analysis is represented by Kiesenbauer (2012), which focuses on the German market. The study distinguishes five product categories (traditional endowment policies, annuities and long-term health contracts, term life insurance, group business and unit-linked contracts), and includes both macroeconomic explanatory variables (e.g. current market yield, DAX performance, gross domestic product and unemployment rate) and company-specific explanatory variables (e.g. company ages, distribution channel, company legal form and company size). One of the most recent study is the working paper Hwang et al. (2014) on Taiwan data from the decade 1999-2009. Just like in Kiesenbauer (2012), the paper considers a number of macroeconomic variables and company-specific variables, that is, business line, premium income, company age, return-on-asset, domestic/foreign company, unemployment rate, home-ownership ratio, short-term interest rate and economic growth rate.
Beyond the huge amount of empirical studies on surrender rates (only partially described so far in this section) trying to detect the most relevant predictors, a remarkable number of studies about surrender option's valuation also exists. Such papers deal with the surrender activity of policyholders as the exercise of an American option embedded in the insurance contract, and evaluate it as a stand-alone option by using either analytic or numeric models. Given that our goal is not product pricing, the topic is beyond our scope, and it will not be analysed further.
Obviously, persistence is a crucial factor in the pure financial market as well.

For instance, the author of Stanton (1995) values mortgage-backed securities in the US assuming that part of the prepayment decision of mortgage holders is rational and based on current economic conditions, while the remaining part is interpreted as irrational. Again, this is beyond our scope, and the topic will be not analysed further.
To sum up, three categories of PHB drivers can be empirically distinguished: macroeconomic factors, company-specific factors and policy-specific factors. Of course, other factors could relate to life insurance surrender activity, although they might not have mentioned in past studies yet. At the same time, the set of relevant explanatory variables could change in time, for example, as target clients, product nature, or insurance purpose change. As a consequence, looking for a unique, stable set of explanatory variables seems to be the wrong way to go ahead. Therefore, we will focus on the dataset provided by a single insurance company, where most of the explanatory variables are policy-specific, while only one macroeconomic variable is included (obviously, there is no reason to include company-specific variables, given that data come from the same company).

4.3 Segregated fund modelling

The crucial aspect of Italian segregated fund modelling is the simulation of the *crediting rate*, that is, the yield used to reevaluate the sum assured (and the reserve) of the policyholder. Such a yield will be a function of the fund rate as well as a number of tariff parameters.
For the economic scenario generation, we will use a Gaussian two-factor model (like in PIA (2015) and Aleandri (2016)), which guarantees a number of useful properties. First, it embeds an instantaneous linear correlation between rates at different maturities, while single-factor models (e.g. CIR, Vasicek, etc.) implicitly assume correlation 1 (see Brigo et al. (2001)): effectively, each simulation leads to a rigid movement of the interest rate curve. Another reason relates to the fitting of the actual interest curve: while single-factor models, indeed, fit it, two-factor models like the Gaussian one can adapt to it perfectly.
The short rate under the Gaussian model is defined by the following equation:

$$i_t := X_t + Y_t + \phi(t) \tag{4.1}$$

where

$$dX_t = -\mu_x X_t dt + \sigma_x dZ_t^x \tag{4.2}$$
$$dY_t = -\mu_y Y_t dt + \sigma_y dZ_t^y \tag{4.3}$$

and initial conditions $X_0 = 0$ e $Y_0 = 0$. The instantaneous linear correlation between X and Y is represented by the parameter ρ:

$$dZ_t^x dZ_t^y = \rho dt. \tag{4.4}$$

parameter	value
μ_x	0,40100
σ_x	0,03780
μ_y	0,17800
σ_y	0,03720
ρ	-0,99600
α_0	0,00044
β_0	-0,31131
β_1	30,00000
β_2	-26,98974
τ_1	0,13474
τ_2	0,16187

Fig. 4.1: Gaussian two-factor model and Nelson-Siegel-Svensson parameters

The parameters in the equations (4.2), (4.3) and (4.4) are the same as in PIA (2015) and Aleandri (2016), based on the market data at 25/11/2016. Assuming that the actual ZCB price curve is interpolated by some polynomial function $\Phi(t)$, it can be proven (see Brigo et al. (2001)) that, if $f(t)$ denotes the instantaneous forward rate in t, that is,

$$f(t) := -\frac{d \ln \Phi(t)}{dt} \tag{4.5}$$

then the deterministic function $\phi(t)$ defined by

$$\begin{aligned} \phi(t) &:= f(t) + \frac{\sigma_x^2}{2\mu_x^2}(1 - e^{-\mu_x t})^2 + \frac{\sigma_y^2}{2\mu_y^2}(1 - e^{-\mu_y t})^2 + \\ &+ \frac{\rho\sigma_x\sigma_y}{\mu_x\mu_y}(1 - e^{-\mu_x t})(1 - e^{-\mu_y t}) \end{aligned} \tag{4.6}$$

guarantees a perfect fitting of the actual interest rate curve. However, notice that the choice of the polynomial function still affects the results. An option is the *Nelson-Siegel-Svensson function* (see Nelson et al. (1987) and Svensson (1994)):

$$f(t) := \alpha_0 + \beta_0\left(\frac{1 - e^{-t\tau_1}}{t\tau_1}\right) + \beta_1\left(\frac{1 - e^{-t\tau_1}}{t\tau_1} - e^{-t\tau_1}\right) + \beta_2\left(\frac{1 - e^{-t\tau_2}}{t\tau_2} - e^{-t\tau_2}\right). \tag{4.7}$$

The short rate defined by (4.1) should be calibrated from the actual risk-free curve. We will use the parameters calibrated in PIA (2015) for the German market (see Figure 4.1), which produce the curve in Figure 4.2. However, we also need a stochastic model for the future bond yields since

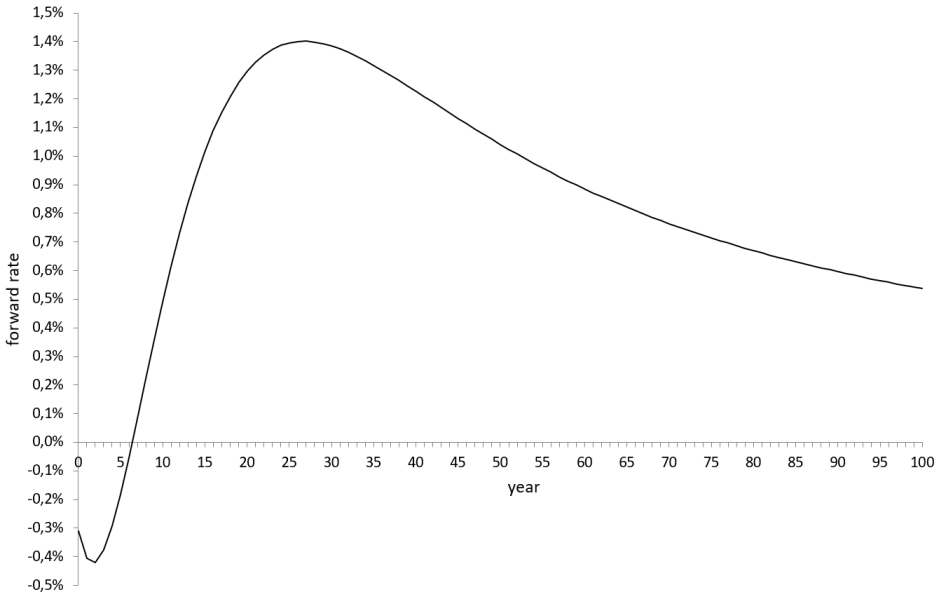


Fig. 4.2: Polynomial function $f(t)$ from the Nelson-Siegel-Svensson model

the segregated fund invests in risky bonds. So we adjust (4.6) as follows:

$$\begin{aligned}\phi^*(t) &:= f(t) + \left(\frac{\sigma_x^2}{2\mu_x^2} + d_x\right)(1 - e^{-\mu_x t})^2 + \left(\frac{\sigma_y^2}{2\mu_y^2} + d_y\right)(1 - e^{-\mu_y t})^2 + \\ &+ \frac{\rho\sigma_x\sigma_y}{\mu_x\mu_y}(1 - e^{-\mu_x t})(1 - e^{-\mu_y t}) \qquad (4.8)\end{aligned}$$

by using two deterministic factors which tends to the parameters d_x and d_y over time. In other words, the bond yield is simulated by the stochastic process

$$r_t := i_t + d_x(1 - e^{-\mu_x t})^2 + d_y(1 - e^{-\mu_y t})^2 \qquad (4.9)$$

or equivalently

$$X_t^* := X_t + d_x(1 - e^{-\mu_x t})^2 \qquad (4.10)$$

$$Y_t^* := Y_t + d_y(1 - e^{-\mu_y t})^2. \qquad (4.11)$$

The parameters d_x and d_y are calibrated in order to match an average 10-year spread approximately equal to the actual 10-year Bund-BTP spread, that is, 186 bps as at 31/12/2017. Specifically, $d_x = d_y = 1,77\%$.
Generally, a minor part of the segregated fund is equity-based, so we need a stochastic model for it as well, for example a classical geometric Brownian motion defined by the risk-free component r_t, the risk premium parameter μ_S and the non-systematic risk parameter σ_S:

$$S_t = S_0 e^{\int_0^t r_\tau d\tau + \left(\mu_S - \frac{\sigma_S^2}{2}\right)t + \sigma_S Z_t^S} \qquad (4.12)$$

where Z_t^S is uncorrelated with both Z_t^x and Z_t^y.
In practice, assume that the segregated fund invests $b \in [0,1]$ asset in n held-to-maturity coupon bonds bought at par and $1-b$ in equity. In $t=0$, when a new insurance contract is underwritten, each bond in portfolio has a different maturity, say $t_1, \ldots, t_n$. In other words, the i^{th} bond pays a known coupon c_i for the next t_i years. Since each bond is bought at par, its annual yield equals the coupon rate itself. More specifically, if F_i denotes the face value of the i^{th} bond, its contribution to the segregated fund rate is

$$C_i := \frac{c_i F_i}{\sum_{k=1}^n F_k} := c_i b_i \tag{4.13}$$

where $\sum_{i=1}^n b_1 = 1$. In fact, as long as $t \leq \min\{t_1, \ldots, t_n\}$, that is, no bond has matured yet, the average yield of the bond component is

$$R_C := \sum_{i=1}^n C_i \equiv \sum_{i=1}^n c_i b_i, \quad \forall t \leq \min\{t_1, \ldots, t_n\} \tag{4.14}$$

which is known in $t=0$ and constant. As soon as the i^{th} bond matures after T_i years, it will be probably replaced by a comparable security, say a new coupon bond with same maturity in T_i years (for practical reasons, assume that T_i is greater than the duration of the insurance contract, in order to replace each bond at most one time). The new par bond yields exactly the stochastic forward rate $f(t_i+1, T_i)$. Using the dummy function $\chi_{t \leq t_i}$, the contribution of the i^{th} bonds to the segregated fund rate in t can be written as follows:

$$C_i(t) := [\chi_{t \leq t_i} c_i + (1 - \chi_{t \leq t_i}) f(t_i+1, T_i)] b_i \tag{4.15}$$

and finally the stochastic return of the whole bond component in the segregated fund:

$$R_C(t) := \sum_{i=1}^n C_i(t) = \sum_{i=1}^n [\chi_{t \leq t_i} c_i + (1 - \chi_{t \leq t_i}) f(t_i+1, T_i)] b_i, \quad \forall t. \tag{4.16}$$

The initial bond component will be represented by three BTPs (i.e., fixed-coupon bonds issued by the Italian government), as shown in Figure 4.3.

Each of them will cover 30% of the entire bond allocation, which equals $b = 90\%$ of the entire asset allocation. In particular, BTP1 has been just bought, BTP2 was bought 10 years ago, and BTP3 was bought 23 years ago. As soon as one of them matures, it will be replaced by a new bond with same maturity and yield equal to the related forward rate from the stochastic scenario.
Furthermore, (4.12) provides the return of the equity component:

$$R_S(t) := \frac{S_t}{S_{t-1}} - 1 = e^{r_{\tau-1} + \left(\mu_S - \frac{\sigma_S^2}{2}\right) + \sigma_S Z} - 1 \tag{4.17}$$

parameter	**BTP1**	**BTP2**	**BTP3**
b_i	30%	30%	30%
c_i	1,0%	3,0%	5,0%
t_i	0 yrs	10 yrs	23 yrs
T_i	10 yrs	15 yrs	30 yrs

Fig. 4.3: Initial bond component parameters of the segregated fund

parameter	**FTSE**	**EURO**	**S&P**
$1-b$	10%		
μ_S	-1,96%	2,46%	9,35%
σ_S	20,22%	12,15%	10,83%

Fig. 4.4: Initial equity component parameters of the segregated fund

where Z denotes a standard normal distribution. For our application, we will consider one three types of equity securities at a time, calibrated on the performances of three market indexes - FTSE MIB, EURO STOXX 50 and S&P 500 - from the period 2010-2016 (see Figure 4.4). Such a component will cover $1-b=10\%$ of the entire asset allocation.
Given the yield contributions $R_C(t)$ and $R_S(t)$ for the bond and equity components respectively, the segregated fund rate is equal to

$$g(t) := bR_C(t) + (1-b)R_S(t). \tag{4.18}$$

Nonetheless, sum assured and reserve revaluation takes into account other contract parameters such as profit sharing η, minimum guaranteed rate ϱ and minimum management fee k. Therefore, the stochastic crediting rate is

$$R(t) := \max\{\min\{\eta g(t), g(t)-k\}, \varrho\} \tag{4.19}$$

assuming no technical rate (which is common in Italian insurance contracts including a minimum rate guarantee).
The insurance contract we will analyse is a deferred capital with duration $n=20$ years, age at entry $x=30$ years, without death benefit and terminal bonus at maturity. Notice that the choice of such a simple contract is due to the necessity of excluding any potential impact from mere technical complexities, in order to better isolate and emphasize the lapse aspect. Nonetheless, the analysis of this chapter can be easily extended to more peculiar products.
Regarding the technical assumptions, we will use the mortality rates q_{x+t} published in ISTAT (2015) for the year 2015, while no lapse rate is included in the calculation of the policy number as we will evaluate the lapse impact *ex post* by averaging on the profit using the estimated lapse probabilities.

As a consequence, the average policy number in t is equal to

$$N_t = \begin{cases} 1 & t = 0 \\ N_{t-1}(1 - q_{x+t}) & \forall t = 1, \ldots, n. \end{cases} \tag{4.20}$$

To simplify the analysis and avoid the allocation of expense reserves, any premium loading is explicit and applied to the annual premium Π, although implicit loadings as well as loadings applied to the reserve are common in life insurance. As a consequence, Π is affected by predetermined acquisition cost α (entirely paid at inception), yearly premium collection cost β and yearly maintenance cost γ. Since Π is also increased by the safety loading l, the total premium paid by the policyholder is

$$\Pi = \frac{P(1 + l)}{1 - \alpha - \beta - \gamma} \tag{4.21}$$

where P is the theoretical fair premium.
Following the local accounting standards, the mathematical reserve in t can be calculated retrospectively as the sum of the mathematical reserve in $t-1$ and the premium paid:

$$V_t = V_{t-1}[1 + R(t)] + P \qquad \forall t = 1, \ldots, n \tag{4.22}$$

where $V_0 = P$. The initial sum assured S_0 is function of P:

$$S_0 = P\frac{\ddot{a}_{\overline{x:n}|}}{{}_nE_x} = P\frac{\sum_{k=0}^{n-1} {}_kp_x}{{}_np_x}. \tag{4.23}$$

where ${}_kp_x$ denotes the k-year survival probability at age x (no deflator appears as no technical rate is anticipated). For any $t > 0$, the sum assured is affected by the increase in reserve due to $R(t)$, so that the following formula for the sum assured holds:

$$S_t = \frac{V_{t-1} + P\ddot{a}_{\overline{x+t:n-t}|}}{{}_{n-t}E_{x+t}} = \frac{V_{t-1} + P\sum_{k=t}^{n-t-1} {}_{k-t}p_{x+t}}{{}_{n-t}p_{x+t}} \qquad \forall t = 1, \ldots, n. \tag{4.24}$$

As an obvious consequence, the rate crediting the sum assured is not $R(t)$, but a much lower rate.
The aforementioned costs are assumed to offset the related expenses generated by the maintenance of the contract, so the initial expense is

$$E_0 = \alpha\Pi \tag{4.25}$$

while the expense in any subsequent t is

$$E_t = (\beta + \gamma)\Pi \qquad \forall t = 1, \ldots, n. \tag{4.26}$$

Since no reserve reduction for acquisition costs is permitted (i.e., the reserve may not be "zillmered"), and no timing gap between maintenance costs and

parameter	value
η	90%
ϱ	1,0%
k	0,2%
n	20 yrs
x	30 yrs
P	1000
l	15%
α	2,0%
β	3,0%
γ	0,5%

Fig. 4.5: Tariff parameters

premium payments occurs, no expense reserve will be allocated.
Once the tariff parameters are determined (see Figure 4.5), we can calculate the annual profit generated by the contract. In $t = 0$, the company receives the first premium and immediately pays the acquisition costs. After that, the company still receives premiums and the related reserve begins to credit at rate $g(t)$; however, maintenance expenses and surrender benefits (from the fourth year) are paid. In the last policy year, the company receives the last return from the reserve, but pays the sum assured. In formulas

$$P\&L_t = \begin{cases} \Pi - E_0 & t = 0 \\ \overline{\Pi}_{t-1} + g(t)\overline{V}_{t-1} - \overline{E}_t - \Delta\overline{V}_{t-1,t} & \forall t = 1, \dots, n-1 \\ g(n)\overline{V}_{n-1} - \overline{E}_n - (\overline{S}_n - \overline{V}_{n-1}) & t = n. \end{cases} \tag{4.27}$$

where $\overline{\Pi}_t := N_t\Pi$, $\overline{V}_t := N_tV_t$, $\overline{E}_t := N_tE_t$, $\overline{S}_t := N_tS_t$ and $\Delta\overline{V}_{t-1,t} := \overline{V}_t - \overline{V}_{t-1}$. As a consequence, the cumulated discounted profit at the end of year t is

$$D_t(\cdot) := \sum_{\tau=1}^{t} P\&L_\tau(\cdot)v_\tau(\cdot). \tag{4.28}$$

where the argument represents some underlying economic scenario, and v_τ denotes the deflator.
As a next step, assume that we can somehow estimate lapse probabilities as at any policy year, given the economic scenario. Let's denote $P_{t-1}(L_t)$ the probability that the policyholder leaves as the end of the policy year t given that he/she did not leave at the end of the policy year $t-1$. Then, for each $t = 1, \dots, n-1$, the unconditioned probability of a lapse at the end of year t is

$$P(t) := P_{t-1}(L_t)\prod_{\tau=1}^{t-1}(1 - P_{\tau-1}(L_\tau)) \tag{4.29}$$

while the unconditioned probability that the policyholder will reach the maturity is

$$P(n) := \prod_{\tau=1}^{n}(1 - P_{\tau-1}(L_\tau)) \tag{4.30}$$

where $P_0(L_1) = P_1(L_2) = P_2(L_3) = 0$ in our specific case.
Given that $\sum_t P(t) = 1$, the average discounted profit over some economic scenario may be calculated as the mean of the potential discounted profit as at the end of each policy year. For instance, if those unconditioned probabilities are denoted with $p_1(\cdot), \ldots, p_n(\cdot)$, the average discounted profit in that scenario will be

$$D(\cdot) := \sum_{t=1}^{n} p_t(\cdot)D_t(\cdot) = \sum_{t=1}^{n} p_t(\cdot)\sum_{\tau=1}^{t} P\&L_\tau(\cdot)v_\tau(\cdot) \tag{4.31}$$

where the argument represents some underlying economic scenario. More specifically, we will consider three type of scenarios, that is, the certainty equivalent, the stochastic scenario with static PHB (i.e., certainty equivalent lapse probabilities) and stochastic scenario with dynamic PHB (i.e., simulation-based lapse probabilities). While the discounted profit of the former is deterministic:

$$D[ce, p(ce)] := \sum_{t=1}^{n} p_t(ce)D_t(ce) = \sum_{t=1}^{n} p_t(ce)\sum_{\tau=1}^{t} P\&L_\tau(ce)v_\tau(ce) \tag{4.32}$$

those of the stochastic approaches depend on the specific simulation generated by the stochastic processes defined earlier in this section. The discounted profits related to the k^{th} scenario with static PHB and dynamic PHB are respectively

$$D[k, p(ce)] := \sum_{t=1}^{n} p_t(ce)D_t(k) = \sum_{t=1}^{n} p_t(ce)\sum_{\tau=1}^{t} P\&L_\tau(k)v_\tau(k) \tag{4.33}$$

$$D[k, p(k)] := \sum_{t=1}^{n} p_t(k)D_t(k) = \sum_{t=1}^{n} p_t(k)\sum_{\tau=1}^{t} P\&L_\tau(k)v_\tau(k) \tag{4.34}$$

and the Monte Carlo proxies of the related stochastic discounted profits are

$$D[N, p(ce)] := \frac{1}{N}\sum_{k=1}^{N} D[k, p(ce)] \tag{4.35}$$

$$D[N, p(N)] := \frac{1}{N}\sum_{k=1}^{N} D[k, p(k)]. \tag{4.36}$$

Additionally, we can consider a stricter lapse definition by setting a full lapse as soon as the estimated probability exceeds the cut-off. In other words,

referring to the k^{th} simulation, the policyholder will decide to lapse at the end of the first policy year t^* whose estimated probability is $p_{t^*}(k) > 50\%$. In this case, the discounted profit over the k^{th} simulation is

$$D[k, t^*(k)] := D_{t^*(k)}(k) = \sum_{\tau=1}^{t^*(k)} P\&L_\tau(k) v_\tau(k) \tag{4.37}$$

and the related Monte Carlo proxy is

$$D[N, t^*(N)] := \frac{1}{N} \sum_{k=1}^{N} D[k, t^*(k)]. \tag{4.38}$$

To summarize, the analysis in Section 4.5 will be based on the average profit over $N = 1000$ simulations for the four aforementioned approaches, that is,

- certainty equivalent scenario with static PHB (i.e., certainty equivalent lapse probabilities) and deterministic profit $D[ce, p(ce)]$
- stochastic scenarios with static PHB (i.e., certainty equivalent lapse probabilities) and stochastic profit $D[N, p(ce)]$
- stochastic scenarios with probability-based dynamic PHB and stochastic profit $D[N, p(N)]$
- stochastic scenarios with time-to-lapse-based dynamic PHB and stochastic profit $D[N, t^*(N)]$.

Although those average discounted profit are unquestionably the reference measures to value an insurance contract, there are other relevant measures as well. In particular, we will also consider the *time value of options and guarantees* (TVOG) calculated as the difference between the certainty equivalent profit and the stochastic average profit in the three different cases:

$$TVOG[N, p(ce)] := D[ce, p(ce)] - D[N, p(ce)] \tag{4.39}$$
$$TVOG[N, p(N)] := D[ce, p(ce)] - D[N, p(N)] \tag{4.40}$$
$$TVOG[N, t^*(N)] := D[ce, p(ce)] - D[N, t^*(N)]. \tag{4.41}$$

TVOG is crucial because it measures the value of any options and guarantees embedded in the contract. In our case, $D[ce, p(ce)]$ is a favourable scenario where the minimum guaranteed rate plays no relevant role. This is the reason why $D[ce, p(ce)]$ is very stable or even constant in most of the plots of Section 4.5. The full effect of the guarantee is only evident in the stochastic profit, that is, in the TVOG component.

4.4 Ensembles in machine learning

Broadly speaking, a machine learning *ensemble* is any combination of predictions from different tools. For example, the tool used in Subsection 3.5.4 to estimate individual case reserves is an example of ensemble. Nonetheless, there is a range of common ensembles which will be introduced in this section. All of them are based on the concept of *bootstrap* (appeared in Efron (1979) for the very first time), that is, repetitively simulating from a model in order to improve the knowledge about the underlying phenomenon. Consider a distribution F and some statistic of F, $T(y_1, \dots, y_n)$, function of n observations, representing a characteristic θ of F (e.g., the mean or the variance). We may want to estimate θ by using bootstrap. If we know the analytical form of F, we can estimate its parameters by maximizing the likelihood, and then simulate independent observations from the estimated distribution $\widehat{F}$. If we don't know the analytical form of F, but we have a population drawn from F, we can assume that its empirical distribution is an estimation for F: we can randomly draw - with replacement - from it, and still obtain a sample of independent observations. Whether F is analytical or empirical, there is always a straightforward way to get an estimation of F, say $\widehat{F}$, and sample from it. More importantly, $\widehat{F}$ converges uniformly to F, as stated by Theorem 20.6 in Billingsley (1995).
Bootstrap is actually based on sampling. We can sample n observations as many time as we want from $\widehat{F}$. Denote the i^{th} sample as $\boldsymbol{y}_i = y_{i1}, \dots, y_{in}$, and calculate the i^{th} sample estimation for θ:

$$\widehat{\theta}_i := T(\boldsymbol{y}_i). \tag{4.42}$$

Repeat the computation for a high number of samples, say N. Finally, the bootstrap estimation of θ is the mean of all the sample estimations:

$$\widehat{\theta} := \frac{1}{N} \sum_{i=1}^{N} T(\boldsymbol{y}_i). \tag{4.43}$$

For instance, if θ represents the mean of F, (4.43) becomes

$$\widehat{\theta} := \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{n} y_{ij} \tag{4.44}$$

which is an alternative to the traditional sample mean as a population mean estimator:

$$\overline{y} := \sum_{k=1}^{n} y_k. \tag{4.45}$$

It is worth noting the resemblance between the bootstrap and Monte Carlo methods: the key concept is basically the same.

As we will easily see in the next subsections, classical ensembles in machine learning are all based on the same bootstrap concept. They are typically used to improve CART performances. Indeed, the main drawback of trees is the high volatility of their predictions. For example, if we use two training datasets from the same database, the two full trees may be quite different. Bootstrap represents a solution.

4.4.1 Bagging

Bootstrap aggregation, or *bagging*, was first proposed in Breiman (1996). It represents the most natural way to reduce variance in a machine learning tool: run it on N random samples, and average the N predictions. This is indeed in equation (4.43). If the variance of $T(\boldsymbol{y}_i)$ is σ^2, then

$$\begin{aligned} Var(\widehat{\theta}) &= Var\left(\frac{1}{N}\sum_{i=1}^{N} T(\boldsymbol{y}_i)\right) = \frac{1}{N^2}\sum_{i=1}^{N} Var(T(\boldsymbol{y}_i)) = \\ &= \frac{1}{N^2}\sum_{i=1}^{N}\sigma^2 = \frac{\sigma^2}{N} \end{aligned} \tag{4.46}$$

which is a reduction in estimation variance by a factor $\frac{1}{N}$. In bagging, we use N random samples - with replacement - from the training dataset (or, ideally, N different training datasets), apply the same machine learning tool on each of them, and average the N different predictions for the j^{th} training record:

$$\widehat{\theta}_j^{bag} := \frac{1}{N}\sum_{i=1}^{N}\widehat{\theta}_{ij}, \qquad \forall j = 1, \ldots, n \tag{4.47}$$

Each bagged tree is a full tree, that is, it is not pruned, so it gets very low bias and very high variance. The latter is however reduced by bagging. Notice that (4.47) may be used for classification problems too, as long as $\widehat{\theta}_j^{bag}$ is rounded to 0 or 1.

Once we get the estimations from the bagging procedure, they can be validated in the usual way by using the validation dataset. However, there is a further, less expensive option. First, consider a random sample from a database with n observations. The probability of drawing the j^{th} observation for the sample is $\frac{1}{n}$, so the probability of not drawing it is $1-\frac{1}{n}$. That's just for the first draw. Since there are a total of n sample, all of which are independent and with replacement, the probability of never choosing the j^{th} observation on any of the draws is $(1-\frac{1}{n})^n$. If n is great enough, that probability approximates to $e^{-1} \approx 36,8\%$. In other words, around $\frac{1}{3}$ training records are not used for the j^{th} bagged prediction. So we can use those *out-of-bag* observations as validation records for the j^{th} bagged tree. Using this approach, we do not even need a standalone validation step on a separated validation dataset. Actually, we may use the entire database as

the training dataset.
The higher the number of bagged trees, the lower the interpretability of predictions. When aggregating trees, we have to give up the nice visual diagram of one single tree. Nonetheless, we can still get an indication of the *variable importance* of each predictor. For classification trees, the variable importance is usually defined as the reduction in misclassification error due to the split over a given predictor, averaged over the number of bagged trees. For regression trees, the variable importance is instead defined as the reduction in the sum of squared errors due to the split over a given predictor, averaged over the number of bagged trees.
Although bagging reduces variance by definition, this reduction is typically small. Indeed, remember that (4.46) holds if the bagged predictions are independent, that is, very different with each other. Unfortunately, a dataset often includes few predictors that explain most of the variance and many, less significant - but still significant - predictors. There is a high chance that most of the bagged trees will be heavily impacted by the former only. As a result, the bagging algorithm will return many similar bagged trees returning similar predictions: this will not reduce variance materially.
Actually, we need smarter aggregation algorithms.

4.4.2 Random forests

Random forests represent a simple enhancement of bagging. Although the algorithm was originally designed to aggregate trees (see Ho (1995)), it can be easily extended to other tools.
Remember that the main drawback of bagging is that the most significant predictors in the dataset will be always used for the first splits in any bagged tree. It results in many, highly correlated bagged trees, which impede a significant reduction in variance. In a random forest, each split in every tree is forced to take into account only a subset of the m predictors. Before any split, the algorithm selects $p < m$ predictors randomly, and performs the split by using one of them. This is actually an easy way to introduce further variance in the process. The random forest can thus generate many different trees, and when the strongest predictors are not selected for the first splits, there is a significant chance that the final tree could not have been generated by the bagging algorithm.
How should we set p? A standard choice in statical software such as R and SAS is $p = \sqrt{m}$ rounded down. This implies a quite low number of possible predictors at each split, guaranteeing a fair variety in the forest. However, there could be reasons to believe that this is not the best choice. Typically, if we have high (respectively: low) correlations among predictors, small (respectively: great) values for p should be more suitable.
An option for a further randomization of random forests is described in Geurts et al. (2006). In a random forest, the subset of predictors is random,

while the splitting rule is defined by minimizing some impurity measure. In *extremely randomized trees*, or *extra trees*, both the subset of predictors and the splitting rule are random, in order to introduce an additional source of variance.

4.4.3 Boosting

Just like bagging and random forests, *boosting* is mainly used to improve performance of CARTs, but it can be easily adapted to any other machine learning tool. However, boosting is not built on bootstrapped samples, although it is still based on trees aggregation. It was informally introduced in Kearns (1988) and analysed in Schapire (1990) a couple of years later.
Bagging and random forests are both based on a rather simple concept: the bigger the sample, the better the estimation. Are there any alternative to improve estimators? In Kearns (1988), the author cites the so-called *hypothesis boosting problem*:

> *Informally, this problem asks whether an efficient learning algorithm* [...] *that outputs an hypothesis whose performance is only slightly better than random guessing implies the existence of an efficient algorithm that outputs an hypothesis of arbitrary accuracy.*

In our context, is there any reason to believe that an ensemble of weak estimators (*weak learners*) may generate a single, strong estimator (*strong learner*)? The paper Schapire (1990) focuses on the hypothesis boosting problem, and proves that

> *a model of learnability in which the learner is only required to perform slightly better than guessing is as strong as a model in which the learner's error can be made arbitrarily small.*

A sequence of weak estimators should thus predict as well as a sequence of bootstrapped estimators. So we can leverage weakness rather than randomness.
There are a great number of boosting algorithms that develop the concepts in Schapire (1990). However, the main idea remains the same. First, set the *interaction depth* d, that is, the number of splits of each tree. Because we want to aggregate weak learners, d should be quite small, for instance $d = 1$ or $d = 2$. Let's use it for a first, weak prediction for the j^{th} training record:

$$\widehat{\theta}_{0j}^{boost} := \widehat{\theta}_{0j} = \theta_j + \epsilon_{1j}, \qquad \forall j = 1, \dots, n \tag{4.48}$$

where ϵ_{1j} is the estimation error from the initial step. Using the same predictors, the algorithm then builds a new tree with d splits to predict ϵ_{1j}, so that the boosting estimator at the subsequent step is

$$\widehat{\theta}_{1j}^{boost} := \widehat{\theta}_{0j} + \widehat{\epsilon}_{1j} = \theta_j + \epsilon_{2j}, \qquad \forall j = 1, \dots, n. \tag{4.49}$$

Once again, the algorithm builds a tree to predict ϵ_{2j}, and obtain the new estimation:

$$\widehat{\theta}_{2j}^{boost} := \widehat{\theta}_{0j} + \widehat{\epsilon}_{1j} + \widehat{\epsilon}_{2j} = \theta_j + \epsilon_{3j}, \qquad \forall j = 1, \dots, n. \tag{4.50}$$

The iteration goes on until the N^{th} estimation:

$$\widehat{\theta}_{j}^{boost} := \widehat{\theta}_{0j} + \sum_{i=1}^{N} \widehat{\epsilon}_{ij}, \qquad \forall j = 1, \dots, n. \tag{4.51}$$

Weak learners are thus used to fit the residuals of previous (weak) estimations rather than the target variable in the usual way.
Notice that this algorithm is quite prone to overfit as N increases. Indeed, we don't know how weak the single trees are, even if d is very small. To better control the learning process, we can additionally define a *learning rate* λ (analogue to that of neural networks in Subsection 3.4.4) and adjust the algorithm so that the first estimation is

$$\widehat{\theta}_{1j}^{boost}(\lambda) := \widehat{\theta}_{0j} + \lambda\widehat{\epsilon}_{1j}, \qquad \forall j = 1, \dots, n \tag{4.52}$$

the second one is

$$\widehat{\theta}_{2j}^{boost}(\lambda) := \widehat{\theta}_{0j} + \lambda\widehat{\epsilon}_{1j} + \lambda\widehat{\epsilon}_{2j}, \qquad \forall j = 1, \dots, n \tag{4.53}$$

and finally

$$\widehat{\theta}_{j}^{boost}(\lambda) := \widehat{\theta}_{0j} + \lambda\sum_{i=1}^{N} \widehat{\epsilon}_{ij}, \qquad \forall j = 1, \dots, n \tag{4.54}$$

for some N, which will be likely higher than before since λ is further slowing the learning process.
Starting from this general algorithm, a number of authors proposed more specific algorithms with various rationales and several applications. The most successful one, *adaptive boosting* (or *AdaBoost*), was introduced in Freund et al. (1997) for classification. For the j^{th} record in the training dataset, initialize a weight $w_{1j} \equiv \frac{1}{n}$, where n is the number of training records. Then, the algorithm builds a first weak learner by minimizing the sum of errors for misclassified records weighted on w_{1j} (at initialization, the errors are equally weighted):

$$\epsilon_1 := \sum_{\widehat{\theta}_j \neq \theta_j} w_{1j} \tag{4.55}$$

which returns the weak estimations $\widehat{\theta}_{1j}$. Set

$$\alpha_1 := \frac{1}{2}\ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right), \tag{4.56}$$

calculate the boosted estimation for the first step:

$$\widehat{\theta}_{1j}^{AdaBoost} := \text{sign}(\alpha_1 \widehat{\theta}_{1j}), \qquad \forall j = 1, \dots, n \tag{4.57}$$

and update the weights for the next one:

$$w_{2j} := w_{1j} e^{-\alpha_1 \theta_j \widehat{\theta}_{1j}}, \qquad \forall j = 1, \dots, n. \tag{4.58}$$

In the second step, the new weak learner returns the weak estimations $\widehat{\theta}_{2j}$ by minimizing the error

$$\epsilon_2 := \sum_{\widehat{\theta}_j \neq \theta_j} w_{2j}. \tag{4.59}$$

Once again, set

$$\alpha_2 := \frac{1}{2} \ln \left(\frac{1 - \epsilon_2}{\epsilon_2} \right) \tag{4.60}$$

calculate the boosted estimation for the second step:

$$\widehat{\theta}_{2j}^{AdaBoost} := \text{sign}(\alpha_1 \widehat{\theta}_{1j} + \alpha_2 \widehat{\theta}_{2j}), \qquad \forall j = 1, \dots, n \tag{4.61}$$

and update the weights for the next one:

$$w_{3j} := w_{2j} e^{-\alpha_2 \theta_j \widehat{\theta}_{2j}}, \qquad \forall j = 1, \dots, n. \tag{4.62}$$

After N steps, the final (strong) estimation for the j^{th} record will be

$$\widehat{\theta}_j^{AdaBoost} := \text{sign} \left(\sum_{i=1}^{N} \alpha_i \widehat{\theta}_{ij} \right), \qquad \forall j = 1, \dots, n. \tag{4.63}$$

Notice that the weights should be renormalized at every step in order to sum up to 1.
Historically, AdaBoost represents the first boosting algorithm that is able to update the weights α_i in an automatic fashion. It is worth mentioning more recent algorithms such as LPBoost, TotalBoost, BrownBoost, xgboost, MadaBoost, LogitBoost and others. All of them are beyond the scope of this dissertation.

4.5 An application to the Italian insurance market

This section describes the path followed to predict the lapse rate from a deferred capital insurance dataset. Using such a dataset, we will be able to predict the lapse probability by policy year. After that, we will use include those estimated probabilities into the simplified ALM model introduced in Subsection 4.3. More specifically, they will define the vector p_k.
However, we do not intend to generalize our results. They are specific for

one particular portfolio, whereas they may turn out to be completely different for other portfolios. Nonetheless, we discussed in Section 4.2 how heterogeneous the lapse's explanatory variables may be, varying by country, business line, or even policyholder. After such a huge number of empirical studies, we do not intend to discuss further the fundamental sources of PHB. By contrast, we will focus our attention on the impact that unanticipated surrender activity can have on the profit of a product. About that, much less has been written, especially because of the difficulties in embedding a comprehensive PHB model into profit valuation.

4.5.1 Data

The results we will show are based on the surrender data provided by an Italian insurer from the period 2005-2006 for part of its endowment business. The raw lapse rate from the dataset is around 10,09%, as already mentioned in Section 4.3. Each of the 11770 records represents the state of a specific tariff as at year end 2005. The dataset includes the following fields:

- `Maturity` for the remaining maturity of the product
- `Sex` for the sex of the policyholder
- `Age` for the age of the policyholder
- `NoPremium` for the number of premium instalments that still need to be paid
- `Premium` for the yearly premium amount
- `AlphaCost` for the yearly acquisition cost percentage on the net premium
- `BetaCost` for the yearly collection cost percentage on the net premium
- `GammaCost` for the yearly maintenance cost percentage on the net premium
- `Reserve` for the actual reserve
- `SumAssured` for the actual sum assured
- `TermBonus` for the actual terminal bonus
- `MinGuarRate` for the yearly minimum guaranteed rate
- `DeltaReturn` for the yearly difference between the one-year forward rate of the Italian sovereign yield curve and the product crediting rate
- `Lapse` for the lapse (yes/no).

It is worth clarifying that, in light of the European Directive 2004/113/CE - better known as "gender directive" - `Sex` may not be used as a pricing feature to differentiate premiums in the European Union. Nonetheless, we will see that `Sex` is a rather irrelevant predictor for PHB (see Figure 4.35, for instance). In any case, given that our analysis will be much more focused on profit valuation than actual pricing, this would not be a problem even if `Sex` was relevant, as long as the results do not impact any policy term (e.g., surrender penalties).
In fact, one single macroeconomic variable is available, `DeltaReturn`, which represents the gap between a reference market rate and the interest guaranteed by the product, just like in several published studies (for example, see Kim (2010)) and in the common actuarial practice. Naturally, we could add other relevant macroeconomic variables among those aforementioned in Section 4.2 (e.g. unemployment rate and gross domestic product), but such variables typically change very smoothly in relation to the actual economic condition. Given that our analysis focuses on data of a specific company, and the historical horizon is not so long, we excluded any pure macroeconomic variable from the analysis.
On the other hand, we can use a number of policy-specific features:

- policyholder-related variables (`Sex` and `Age`)
- tariff-related variables (`Maturity`, `NoPremium`, `Premium`, `AlphaCost`, `BetaCost`, `GammaCost` and `MinGuarRate`)
- path-dependent variables (`Reserve`, `SumAssured` and `TermBonus`).

The distinction between tariff-related variables and path-dependent variables is fundamental to understand how the dataset has been structured.
As confirmed in several empirical studies (for example, see Stanton (1995)), surrender activity of a single policyholder - whether rational or irrational - tends to depend on a limited period of time. In other words, it is unlikely that policyholders will base their decisions on what has happened many years ago, or what is going to happen in many years. This leads to a PHB depending on the current year's condition only, that is, current age, current number of premium payments, current duration, current reserve, current sum assured and difference between the current reference market rate and the current crediting rate.
In Figure 4.6, we reported statistics and correlations calculated from the features in the dataset. As expected, skewness and kurtosis of currency-based variables such as `Premium`, `Reserve`, `SumAssured` and `TermBonus` are very high. At the same time, means and standard deviations differs a lot because of the different nature of each feature. To make the dataset more homogeneous and avoid scale distortions, we first replace each currency field with its natural logarithm, and then standardize the whole dataset. The resulting statistics and correlations are reported in Figure 4.7 (`Premium`, `Reserve`,

	Maturity	Sex	Age	NoPremium	Premium	AlphaCost	BestaCost	GammaCost	Reserve	SumAssured	TermBonus	MinGarRate	DeltaReturn	Lapse
mean	21,31	0,41	51,48	16,51	2.393,55	3,78	5,01	1,21	52.273,11	41.580,83	3.882,14	2,93	0,03	**0,10**
st. deviation	11,31	0,49	11,10	12,30	11.040,81	0,26	0,79	0,94	47.952,66	43.621,49	3.903,58	0,91	1,17	**0,30**
skewness	0,82	0,39	-0,14	0,70	41,97	-0,40	-0,35	0,07	9,26	9,53	6,65	-0,88	0,02	**2,65**
kurtosis	0,38	-1,85	-0,36	0,15	1.921,21	-1,69	0,16	-0,97	163,54	170,26	96,54	-0,31	-1,15	**5,02**

	Maturity	Sex	Age	NoPremium	Premium	AlphaCost	BestaCost	GammaCost	Reserve	SumAssured	TermBonus	MinGarRate	DeltaReturn	Lapse
Maturity	100,0%													
Sex	4,9%	100,0%												
Age	-68,6%	-2,3%	100,0%											
NoPremium	83,2%	3,8%	-56,4%	100,0%										
Premium	-10,1%	-3,3%	8,0%	-12,2%	100,0%									
AlphaCost	16,2%	4,7%	-10,0%	22,7%	3,7%	100,0%								
BestaCost	6,7%	2,1%	-5,9%	6,3%	-1,4%	-30,6%	100,0%							
GammaCost	21,8%	5,7%	-14,4%	27,1%	3,3%	76,1%	-32,2%	100,0%						
Reserve	-16,8%	-13,6%	11,7%	-4,3%	56,9%	-4,3%	1,1%	-5,5%	100,0%					
SumAssured	-12,9%	-12,8%	9,2%	-0,3%	57,2%	3,7%	0,6%	3,2%	98,5%	100,0%				
TermBonus	-24,4%	-15,3%	16,4%	-11,2%	48,3%	-22,6%	5,9%	-24,8%	94,3%	90,0%	100,0%			
MinGarRate	-13,8%	-2,1%	9,7%	-13,7%	1,3%	-37,4%	18,4%	-39,8%	16,9%	15,9%	26,4%	100,0%		
DeltaReturn	-0,6%	0,3%	0,2%	-1,2%	1,3%	0,4%	0,0%	1,1%	1,1%	1,1%	1,3%	0,2%	100,0%	
Lapse	**-4,4%**	**-2,3%**	**5,4%**	**-4,1%**	**0,5%**	**1,9%**	**7,5%**	**1,5%**	**8,2%**	**8,8%**	**7,3%**	**1,5%**	**8,7%**	**100,0%**

Fig. 4.6: Descriptive statistics of lapse rate's explanatory variables before standardization

	Maturity	Sex	Age	NoPremium	LnPremium	AlphaCost	BestaCost	GammaCost	LnReserve	LnSumAssured	LnTermBonus	MinGarRate	DeltaReturn	Lapse
mean	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**0,10**
st. deviation	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	**0,30**
skewness	0,82	0,39	-0,14	0,70	0,86	-0,40	-0,35	0,07	-0,66	-1,32	-0,67	-0,88	0,02	**2,65**
kurtosis	0,38	-1,85	-0,36	0,15	3,02	-1,69	0,16	-0,97	1,51	2,65	0,69	-0,31	-1,15	**5,02**

	Maturity	Sex	Age	NoPremium	LnPremium	AlphaCost	BestaCost	GammaCost	LnReserve	LnSumAssured	LnTermBonus	MinGarRate	DeltaReturn	Lapse
Maturity	100,0%													
Sex	4,9%	100,0%												
Age	-68,6%	-2,3%	100,0%											
NoPremium	83,2%	3,8%	-56,4%	100,0%										
LnPremium	-48,5%	-12,3%	31,8%	-50,0%	100,0%									
AlphaCost	16,2%	4,7%	-10,0%	22,7%	7,9%	100,0%								
BestaCost	6,7%	2,1%	-5,9%	6,3%	1,7%	-30,6%	100,0%							
GammaCost	21,8%	5,7%	-14,4%	27,1%	7,1%	76,1%	-32,2%	100,0%						
LnReserve	-18,0%	-18,0%	11,5%	6,9%	49,9%	-4,5%	6,0%	-7,4%	100,0%					
LnSumAssured	-12,1%	-16,3%	7,6%	17,0%	48,8%	8,8%	9,1%	6,6%	94,9%	100,0%				
LnTermBonus	-25,9%	-18,9%	16,9%	-1,4%	44,0%	-20,5%	8,3%	-26,8%	92,7%	88,9%	100,0%			
MinGarRate	-13,8%	-2,1%	9,7%	-13,7%	1,1%	-37,4%	18,4%	-39,9%	21,2%	15,5%	30,7%	100,0%		
DeltaReturn	-0,6%	0,3%	0,2%	-1,2%	2,1%	0,4%	-0,1%	1,1%	1,0%	0,7%	0,7%	0,2%	100,0%	
Lapse	**-4,4%**	**-2,3%**	**5,4%**	**-4,1%**	**11,6%**	**1,9%**	**7,5%**	**1,5%**	**9,5%**	**9,3%**	**8,1%**	**1,5%**	**8,7%**	**100,0%**
t-statistic	**4,79**	**2,45**	**5,81**	**4,42**	**12,67**	**2,01**	**8,11**	**1,63**	**10,37**	**10,08**	**8,82**	**1,60**	**9,50**	**-**

Fig. 4.7: Descriptive statistics of lapse rate's explanatory variables after standardization

`SumAssured` and `TermBonus` have been replaced by `LnPremium`, `LnReserve`, `LnSumAssured` and `LnTermBonus`, while the field names have not been modified to indicate standardization), where some further multicollinearity issues have been highlighted, that is, all the correlations higher than 50%.
All in all, some of the relationships outlined in various empirical papers recalled in Section 4.2 are somehow confirmed by the correlations in Figure 4.7. Maturity is inversely correlated, that is, policyholders tend to lapse in the first years, rather than in the last ones (the premium payment period is often close to the maturity, so its correlation is similar). Premium amount is significantly correlated, although it seems partially due to the loading components, especially beta costs. It can be seen as an instance of the Emergency Fund Hypothesis described in Section 4.2: policyholders lapse for reasons merely related to the expenses, which they can no longer afford. Similarly, both sum assured and reserve are positively correlated with the surrender activity (the two correlations are also very close, given that the reserve is a function of the sum assured): policyholders lapses only when it is really worth it. Finally, policyholders seem to be sensitive to the difference between what the financial market yields and what their policy effectively returns. It can be seen as an instance of the Interest Rate Hypothesis described in Section 4.2: policyholders may also lapse for more rational reasons to other advantageous yields available on the market.
Those are the most significant correlations, but some non-significant correlations are interesting as well. For example, the technical rate guaranteed by the insurer has no significant impact on the surrender activity: it indirectly confirms the policyholder's short-term sight, which ignores the future benefits represented by a higher guaranteed rate.
As explained in Subsection 1.4.1, datasets are usually partitioned to allow for training, validation and test in three separated steps. In classification problems, however, it is often the case that the "interesting" outcome of the target variable - `Lapse` = 1 - is quite rare, causing little "interesting" information. If we partition the dataset randomly, we will further lose information in the training dataset, and the model could be potentially trained in a distorted or incomplete way. This is the reason why *oversampling* is common practice in data mining: we build the training dataset in such a way that lapse occurrences are as likely as non-lapse occurrences (both 50%), and let the validation dataset include all the other records. Obviously, we will have fewer lapse occurrences to validate the algorithm, which is however trained on much more relevant information. We will use this approach in the application of this chapter.
Finally, we will use the following fields: `Maturity`, `Sex`, `Age`, `NoPremium`, `LnPremium`, `AlphaCost`, `BetaCost`, `GammaCost`, `LnReserve`, `LnSumAssured`, `LnTermBonus`, `MinGuarRate`, `DeltaReturn` and `Lapse` (besides an ID variable to identify the different records).

4.5.2 Lapse prediction

In our framework, lapse prediction means using the predictors identified in Subsection 4.5.1 as inputs for some machine learning tool to return an estimation of the lapse probability (`Lapse`). We will start by using the standard classification tree, and then we will switch to more stable methods such as logistic regression, bagging, random forest and boosting.
CARTs were described in Subsection 3.4.3. Full classification trees can be actually built until the very last leaf, because they separate records into two classes only. By contrast, full regression trees could not be built since the algorithm would tend to allocate each single record to one specific leaf, which is useless. This is the reason why one can always try to build the full classification tree, which will classify training records perfectly. In our case, the algorithm leads to the misclassification error by number of splits in Figure 4.10 (for sake of clarity, only the first 40 splits have been reported). You can see the typical shape of the validation error against that of the training error. While the latter constantly decreases as expected, the former is rather stable and does not show relevant decrease after the first split. As a consequence, the best pruned tree is identified by the first split (black dashed line in Figure 4.10), while the minimum error tree is identified by the first 19 splits (grey dashed line in Figure 4.10). The difference in error is around 1%, but the misclassification error is very high for both, that is, 43,5% and 42,5% respectively. Actually, they are not so far from the random model error, which is 50% (in other words, if we try to classify the training records just by chance, the error will converge to 50%, which is the proportion of lapse and non-lapse in the oversampled training dataset). Such a problem is even more evident looking at the ROC curves (Figure 4.11 and 4.12) and the decile charts (Figure 4.13 and 4.14). While the training AUC is very close to 100% (probably, it is not exactly 100% because of few records sharing the same features but different lapse decision), the validation AUC is around 54%. This is a very poor result in terms of accuracy as well as stability, but it is not so surprising since it represents a typical drawback of CARTs. In fact, trees are too dependent on the characteristics and the structure of the underlying data such as few extreme outliers, order of the records, predictors with close relevance, inconvenient pruning and so on. This rigidity might lead to degenerate or almost-degenerate results like ours.
The easiest way to fix it all at once is represented by logistic regression. Traditional linear correlations between the target variable and the predictors are formally useless in this case, but that is still based on a linear regression model. To some extent, we can still assume that the higher the correlations, the better the fitting. Figure 4.7 includes the full correlation matrix. Regarding the correlations with `Lapse`, all of them seem relatively low, but the related t-statistics (last red row in Figure 4.7) reveals that most

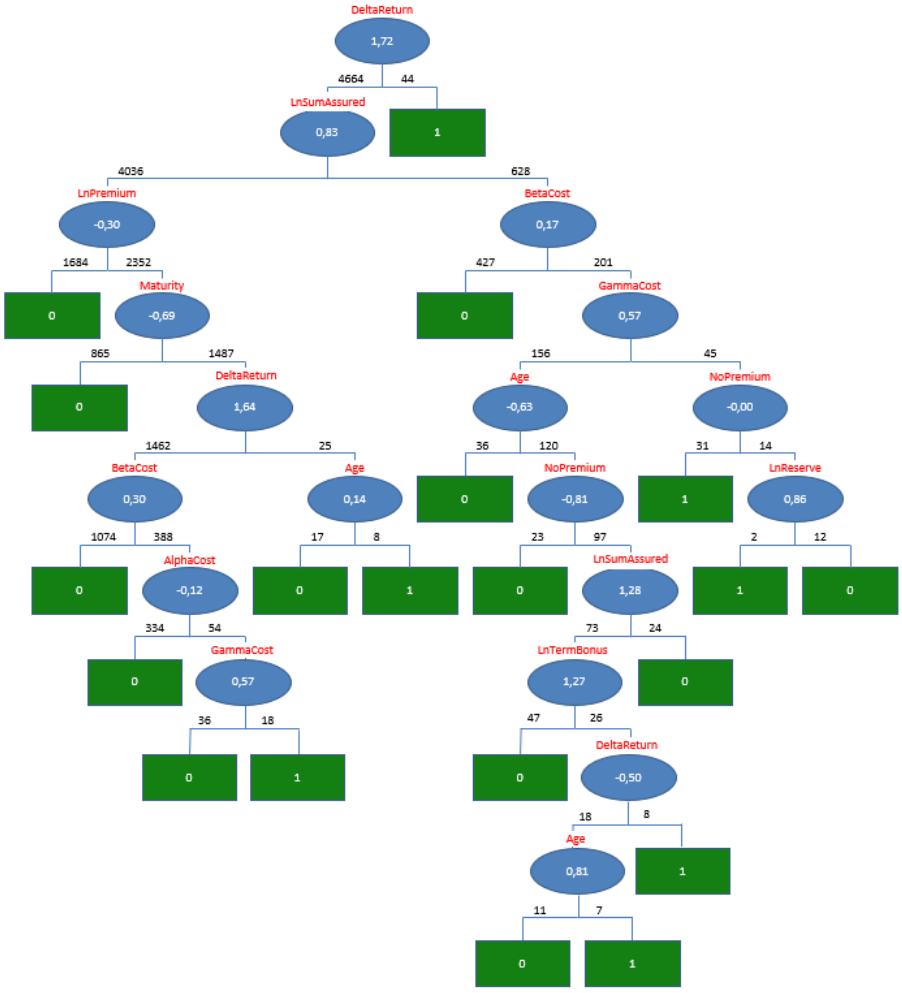


Fig. 4.8: Minimum error classification tree

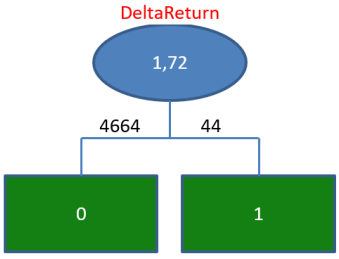


Fig. 4.9: Best pruned classification tree

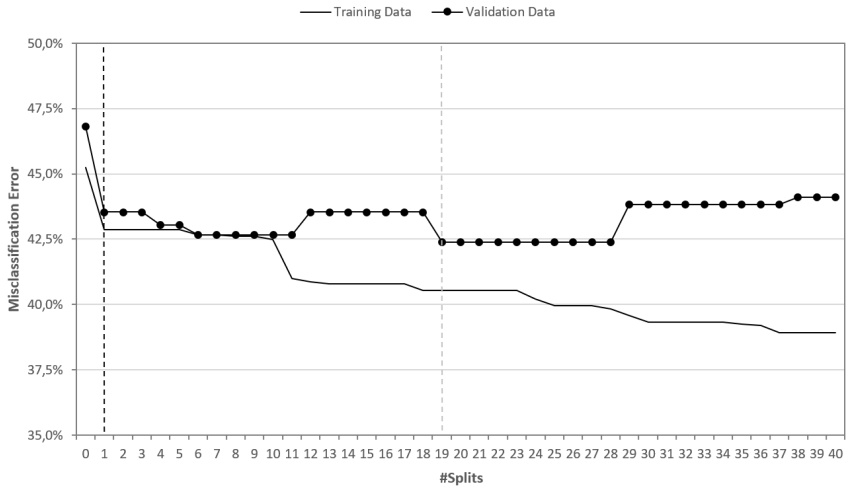


Fig. 4.10: Classification tree - training and validation error varying by number of splits

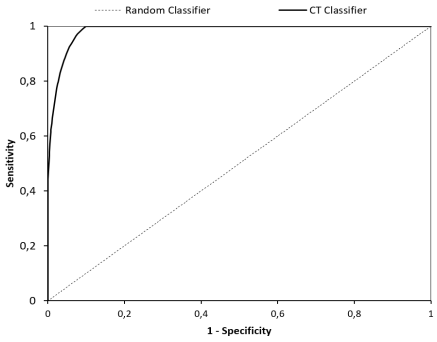


Fig. 4.11: Lapse ROCs for CT (training dataset)

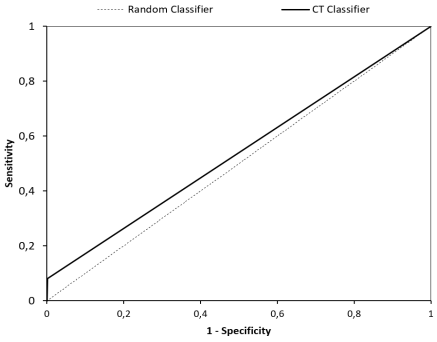


Fig. 4.12: Lapse ROCs for CT (validation dataset)

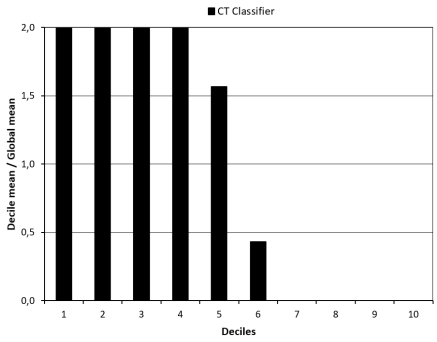


Fig. 4.13: Lapse deciles for CT (training dataset)

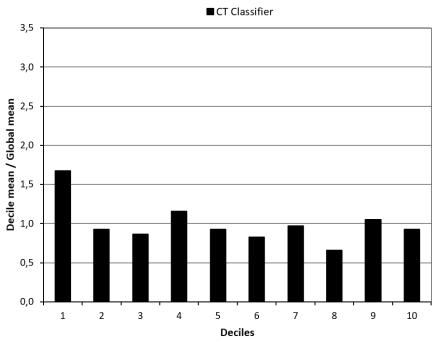
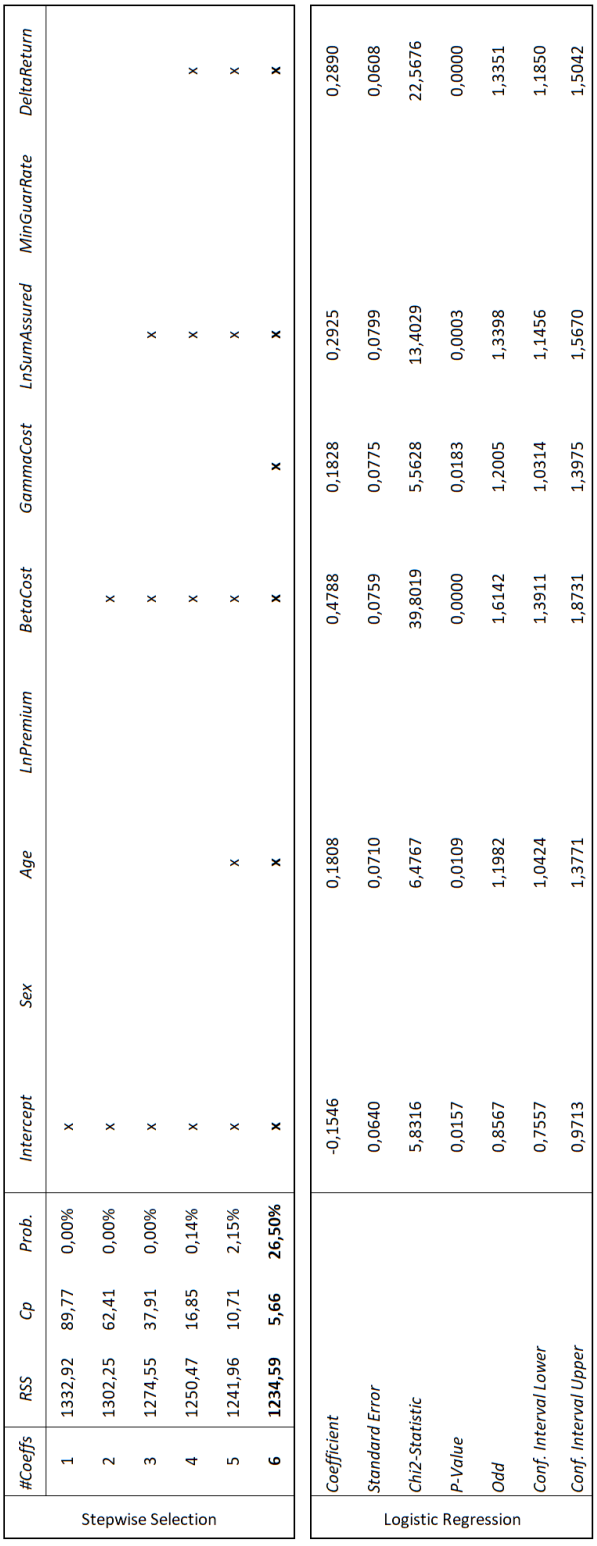


Fig. 4.14: Lapse deciles for CT (validation dataset)

of them - those with t-statistics greater than 2,575 - are significant at 99% confidence level. At the same time, however, there is a number of strong multicollinearities that should be reduced as much as possible before the regression. Therefore, we exclude `Maturity`, `NoPremium`, `AlphaCost`, `LnReserve` and `LnTermBonus` in order to avoid the highest correlations among predictors (yellow cells in Figure 4.7).
Besides the intercept, the stepwise selection kept `Age`, `BetaCost`, `GammaCost`, `LnSumAssured` and `DeltaReturn`. Notice that `GammaCost` does not seem to be significantly correlated to the lapse decision in Figure 4.7, but it is selected in the last step in Figure 4.15. This is probably due to the theoretical misalignment between the linear correlation coefficient and the logistic regression. Moreover, notice that `LnPremium` is not included although it is the most correlated predictor to the lapse decision (see Figure 4.7). The stepwise logistic regression returns the results in Figure 4.15. The explanation is within the correlation matrix once again: `LnPremium` is very correlated with `Age` and `LnSumAssured` (among others). We could have taken them out before the regression, but the stepwise selection itself recognized it. As soon as `Age` and `LnSumAssured` are selected by the model, `LnPremium` will lose most of its predictive power, leading to its rejection.
All in all, the logistic regression returns a training AUC around 69% and a validation AUC around 65%, which are significantly different to those of the classification tree. As expected, logistic regression is much less prone to overfitting and distortions, leading to a worse training performance, but a better validation performance.
After logistic regression, we want to use the machine learning tools introduced in Section 4.4. Both bagging and random forests reduce variance and overfitting by bootstrapping rather than pruning. Bootstrap works around two major drawbacks of CARTs: the data dependency and the pruning performance.
Figure 4.16 shows the convergence of the bagging algorithm over the first 50 weak learners. It converges to a training AUC around 80% and a validation AUC around 72%, which are significantly greater than the logistic regression AUCs. This can be due to a number of reasons. First and foremost, there could be nonlinear relationships caught by the bagging algorithm but ignored by the logistic regression. A second possibility may be an excessive conservatism in variable selection: indeed, the logistic regression used only six predictors, while the bagging included them all. However, we should observe that most of the rejected variables demonstrated multicollinearity issues, which cannot be accepted in a regression framework. Further, the stepwise selection excluded uncorrelated variables such as `Sex` and `MinGuarRate` (see Figure 4.7), so they could hardly explain a significant part of 7% gap in AUC if included in the model.
Once the convergence has been observed, we should choose the number of weak learners to keep. It can be done in several ways as long as they are not



Stepwise Selection	#Coeffs	RSS	Cp	Prob.	Intercept	Sex	Age	LnPremium	BetaCost	GammaCost	LnSumAssured	MinGuarRate	DeltaReturn
	1	1332,92	89,77	0,00%	x								
	2	1302,25	62,41	0,00%	x				x				
	3	1274,55	37,91	0,00%	x				x		x		
	4	1250,47	16,85	0,14%	x				x		x		x
	5	1241,96	10,71	2,15%	x		x		x		x		x
	6	**1234,59**	**5,66**	**26,50%**	**x**		**x**		**x**	**x**	**x**		**x**

Logistic Regression							
Coefficient	-0,1546	0,1808	0,4788	0,1828	0,2925		0,2890
Standard Error	0,0640	0,0710	0,0759	0,0775	0,0799		0,0608
Chi2-Statistic	5,8316	6,4767	39,8019	5,5628	13,4029		22,5676
P-Value	0,0157	0,0109	0,0000	0,0183	0,0003		0,0000
Odd	0,8567	1,1982	1,6142	1,2005	1,3398		1,3351
Conf. Interval Lower	0,7557	1,0424	1,3911	1,0314	1,1456		1,1850
Conf. Interval Upper	0,9713	1,3771	1,8731	1,3975	1,5670		1,5042

Fig. 4.15: Regression summary after stepwise selection

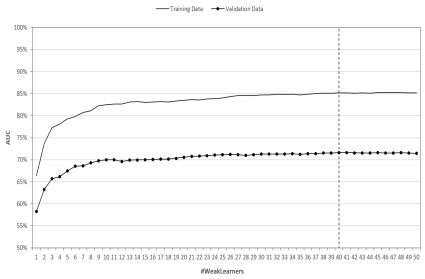


Fig. 4.16: AUC convergence for the bagging algorithm

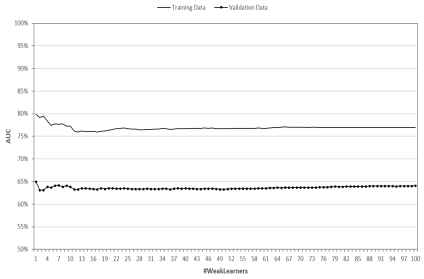


Fig. 4.17: AUC convergence for the random forest

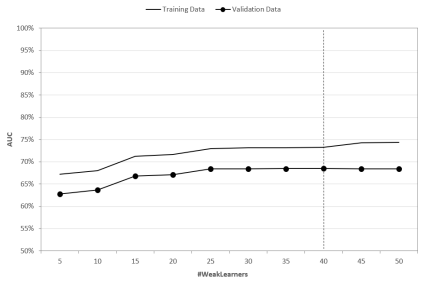


Fig. 4.18: AUC convergence for the AdaBoost at depth 1

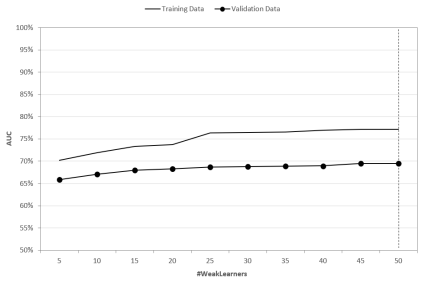


Fig. 4.19: AUC convergence for the AdaBoost at depth 2

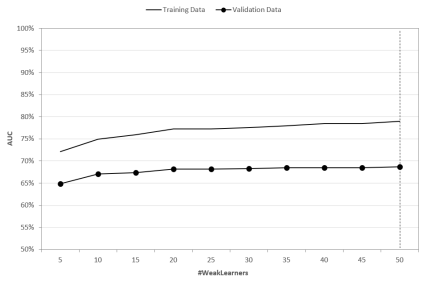


Fig. 4.20: AUC convergence for the AdaBoost at depth 3

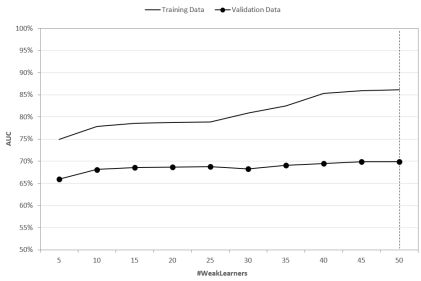


Fig. 4.21: AUC convergence for the AdaBoost at depth 4

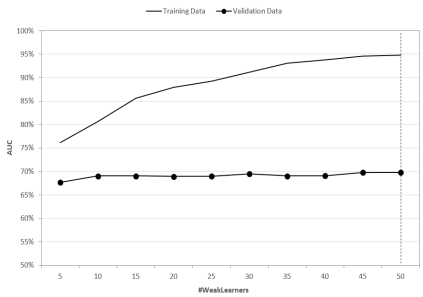


Fig. 4.22: AUC convergence for the AdaBoost at depth 5

so many to cause computational issues. For sake of simplicity, we choose the number of weak learners with the highest AUC, that is, 40 (black dashed line in Figure 4.16).
Such a number of weak learners will be also used as a starting point for the random forest. Recall from Subsection 4.4.2 that the difference between bagging and random forest is that the latter introduce randomness in the choice of predictors. We will bag 40 weak learners for 100 series of random predictors (it should guarantee the convergence just like before), producing a total of 400 weak learners. As explained in Subsection 4.4.2, the number of predictors is generally set equal to the square root of the number of all the predictors in the dataset, that is, $\sqrt{13} \simeq 4$. Figure 4.17 shows the convergence of the random forest algorithm over the 100 series of random predictors. The training AUC and the validation AUC converge to about 77% and 64% respectively. In particular, the latter is much more in line with the logistic regression's training AUC. The random forest cannot outperform the bagging trees because there is likely no "hidden" tree configuration besides that from the bagging algorithm. In other words, the random forest performance is strongly affected by all the 4-predictor series that do not include the most important predictors. As a result, we get a slightly lower accuracy as compared to the logistic regression and even a significantly higher variance as the gap between the training AUC and the validation AUC: 4% for the logistic regression against 13% for the random forest.
After the random forest, we will finally apply the boosting algorithm in its most common form for classification, that is, AdaBoost (see Subsection 4.4.3). It is not a bootstrap-based method, rather it assigns higher weights (i.e., priority) to the misclassified records at each step, and the following weak learner focuses especially on those. While the "weakness" of a random forest's learners is due to the reduction of predictors, the "weakness" of the boosting's learners is due to the reduction of their depth. Specifically, we will limit the growth until the k^{th} level, where $k = 1, \dots, 5$. Just like for the bagging algorithm, we will consider 50 weak learners at most. Figures 4.18-4.22 demonstrate the convergence of the method regardless of the depth. As expected, the greater the depth, the greater the accuracy on the training dataset. However, what is much more interesting is that the validation AUC does not seem related to the depth. Whatever the chosen depth, the validation AUC converges to about 70%, although lower depth could make the convergence slower, for instance, in Figure 4.18 or 4.20). In our case, the validation AUC at the 50^{th} weak learner generally returns the best accuracy, except at depth 1 (see Figure 4.18). The greatest validation AUC corresponds to the 50-weak-learner ensemble at depth 5, so we will use it for comparison with other methods. However, this is only one of the possible choices: given that the same accuracy is shared by less deep and/or less numerous ensembles, which are computationally advantageous, we may

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	403	191	594	191	32,15%
1	239	355	594	239	40,24%
Total			**1.188**	**430**	**36,20%**

Fig. 4.23: Lapse classification for LR (training dataset)

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	6.522	3.466	9.988	3.466	34,70%
1	257	337	594	257	43,27%
Total			**10.582**	**3.723**	**35,18%**

Fig. 4.24: Lapse classification for LR (validation dataset)

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	465	129	594	129	21,72%
1	153	441	594	153	25,76%
Total			**1.188**	**282**	**23,74%**

Fig. 4.25: Lapse classification for BT (training dataset)

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	6.578	3.410	9.988	3.410	34,14%
1	216	378	594	216	36,36%
Total			**10.582**	**3.626**	**34,27%**

Fig. 4.26: Lapse classification for BT (validation dataset)

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	444	150	594	150	25,25%
1	216	378	594	216	36,37%
Total			**1.188**	**366**	**30,81%**

Fig. 4.27: Lapse classification for RF (training dataset)

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	6.428	3.560	9.988	3.560	35,64%
1	271	323	594	271	45,66%
Total			**10.582**	**3.831**	**36,20%**

Fig. 4.28: Lapse classification for RF (validation dataset)

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	517	77	594	77	12,96%
1	83	511	594	83	13,97%
Total			**1.188**	**160**	**13,47%**

Fig. 4.29: Lapse classification for ABT (training dataset)

Actual Class	Predicted Class 0	Predicted Class 1	Cases Number	Errors Number	Errors %
0	6.442	3.546	9.988	3.546	35,50%
1	225	369	594	225	37,88%
Total			**10.582**	**3.771**	**35,64%**

Fig. 4.30: Lapse classification for ABT (validation dataset)

easily choose lower depth and/or number of weak learners.
As a conclusion to this subsection, we will summarize the results obtained from the different methods. Figures 4.23-4.30 report the classification matrices for each of them, both training and validation step. In this case, we do not need to align the cut-off values since the dataset has been oversampled, so the natural choice is always 0,5. On the training dataset, the performance of the logistic regression is clearly poorer than the others. Actually, the advantage of these ensembles is their extreme flexibility to catch any type of dependency among variables. However, part of this variance is often noisy information, so that it will be probably lost in the validation step. This is the reason why there is always a remarkable gap between the training performance and the validation performance in each of the ensembles, while such a gap is negligible in logistic regression. Of course, stability is the major benefit of regression methods, which seldom experience significant overfitting, unless no variable selection is performed even if many variables bring very little information.
On the other hand, some ensembles seem to work better when it comes with accuracy. Although the false-positive percentages (as well as the overall misclassification error) on the validation dataset are quite close with each other - about 34%-36% - notice that the false-negative error can be quite different. It changes from 45,66% of the random forest to the 36,36% of the bagging algorithm. This is exactly what causes the gaps in ROC curves and decile charts in the validation step. The plots are reported in Figures 4.31-4.34: the logistic regression performance seems comparable to the random forest performance as much as the bagging performance seems comparable to the boosting performance. In particular, the decile chart in Figure 4.34 demonstrates that the logistic regression and the random forest predict lapses about 2,25 times more often than a random method, while the bagging and the boosting algorithms predict lapses nearly 3,5 times (on average) more often than a random method.
However, notice the evident differences in the training ROC curves (see Figure 4.31): it is not surprising that the best accuracy relates to the less stable methods. From this perspective, we should certainly choose the bagging trees rather than the boosting trees, if we want to save the improvement in accuracy avoiding high instability levels.
A last relevant result relates to the variable importance, which is defined by the percentage of reduction in misclassification error thanks to the inclusion of each variable in the ensembles. That is reported in Figure 4.35 for bagging, random forest and boosting (logistic regression is not included because of its different conceptual nature and the preliminary variable selection, which make any comparison harder - if not even insignificant), which seem to be relatively aligned. Moreover, it is worth comparing the Figures 4.7 and 4.15 with that variable importance. First of all, notice that the variables selected by the stepwise algorithm in Figure 4.15 are some of the most important

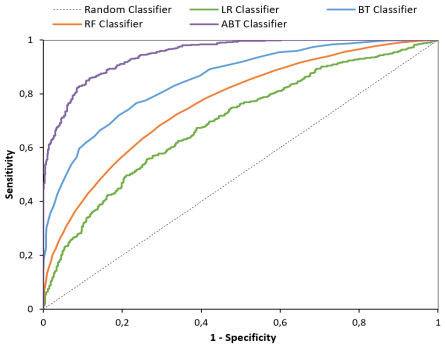


Fig. 4.31: Lapse ROCs (training dataset)

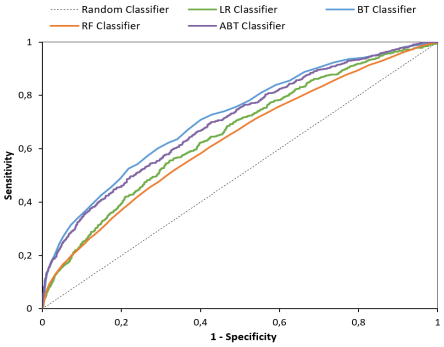


Fig. 4.32: Lapse ROCs (validation dataset)

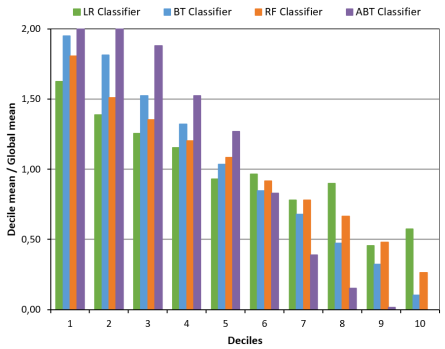


Fig. 4.33: Lapse deciles (training dataset)

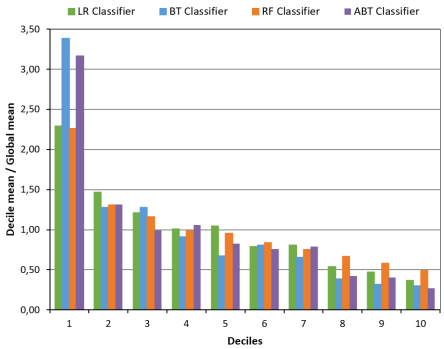


Fig. 4.34: Lapse deciles (validation dataset)

ones, except for `GammaCost`, which is however the less significant with p-value 0,0183. As expected, there are a number of predictors excluded by the regression in spite of their high correlation with `Lapse`, but considered important by the ensembles. They include `Maturity`, `NoPremium`, `LnPremium`, `LnReserve` and `LnTermBonus`. Unsurprisingly, the most important variable is the (natural logarithm of) premium amount, `LnPremium`, which indeed corresponds to the highest correlation with `Lapse`.
All in all

- the logistic regression guarantees the highest level of stability
- bagging and boosting lead to more accurate predictions
- boosting causes more severe instability and overfitting than bagging

so we will use the lapse prediction provided by the bagging algorithm for our profit analysis.

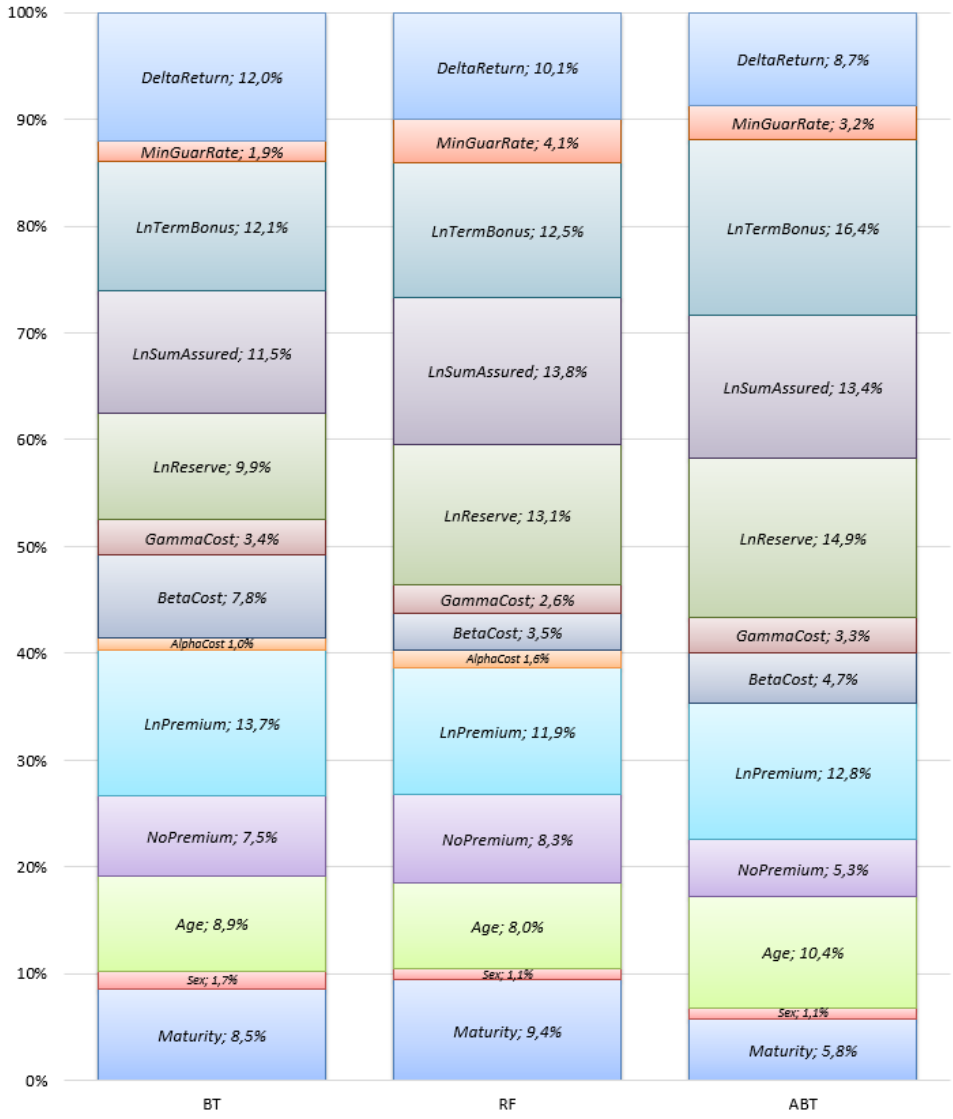


Fig. 4.35: Variable importance percentages

4.5.3 Profit analysis

Using bagging trees to predict lapses has been already suggested in other studies such as Loisel et al. (2011) and Jamal (2016), although they were based on different datasets. A wide range of results can be analysed by integrating the predictions returned by the bagging algorithm and the asset-liability model described in Subsection 4.3. In this section, we try to quantify the impact of dynamic PHB modelling on a traditional deferred capital tariff from the Italian insurance market.

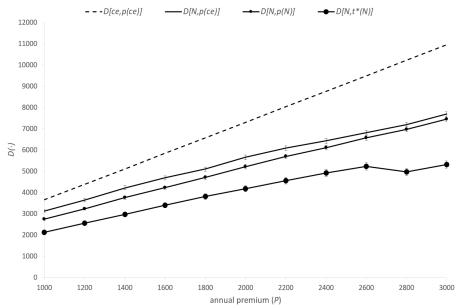


Fig. 4.36: Profit metrics by annual fair premium (EURO STOXX 50 case)

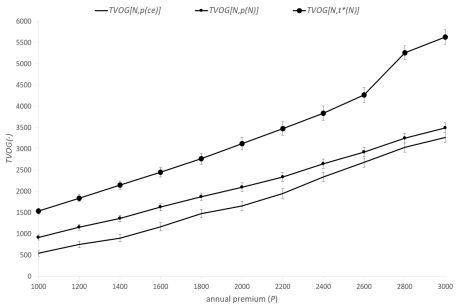


Fig. 4.37: TVOG metrics by annual fair premium (EURO STOXX 50 case)

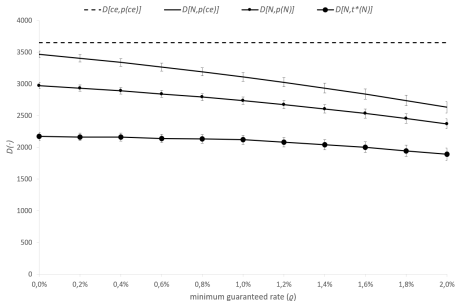


Fig. 4.38: Profit metrics by minimum guaranteed rate (EURO STOXX 50 case)

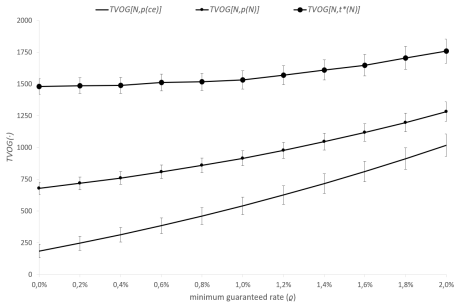


Fig. 4.39: TVOG metrics by minimum guaranteed rate (EURO STOXX 50 case)

The calculation process will be split as follows:

1. calculation of the predictors by scenario simulation
2. lapse prediction through the bagging algorithm by scenario simulation
3. calculation of the lapse-dependent profit by scenario simulation

which are indeed independent with each other.
The results come from $N = 1000$ economic scenarios based on the stochastic model in (4.1), together with the related bond and equity scenarios. The initial set of parameters encompasses the assumptions in Figures 4.1, 4.3, 4.4 and 4.5. Profit and TVOG will be analysed varying by some of those parameters, that is, annual fair premium (P), minimum guaranteed rate (ϱ), initial average coupon (c) by equity investment and equity percentage ($1 - b$) by equity investment.
A first relevant analysis should involve the policyholder reaction to an increase in premium, one of the most important lapse risk factors according

to Figure 4.35. Remember that the reserve is proportional to the sum assured, which is proportional to the premium, that is, an increase in premium will impact two further important variables such as `LnReserve` and `LnSumAssured` (`LnTermBonus` would be also impacted, if it were not zero). Figure 4.36 compares $D[ce, p(ce)]$, $D[N, p(ce)]$, $D[N, p(N)]$ and $D[N, t^*(N)]$ when the annual premium varies from the initial 1000 to 3000. Take into account that the increase primarily affects the fair premium, so the final impact on the annual premium payments, which are impacted by loading and costs, is even greater. Also, the increase in fair premium leads to a proportional increase in the initial sum assured and thus in the reserve.
Basically, an increase in premium should increase the profit, and this is exactly what happens. However, $D[N, p(ce)]$ and $D[N, p(N)]$ are quite closed to each other all along their increment, suggesting that the probabilistic approach does not lead to a strong impact of the dynamic PHB. On the contrary, such an impact is more visible with $D[N, t^*(N)]$, and $TVOG[N, t^*(N)]$ grows much faster than $TVOG[N, p(ce)]$ and $TVOG[N, p(N)]$ in Figure 4.37. As long as the premium increase remains tolerable - around 2600 - the surrender activity shows up smoothly, and $D[N, t^*(N)]$ keeps increasing, but after that threshold the policyholder reaction is so immediate to offset the profit of the contract. Probably, in case of better economic conditions, the policyholder tolerance would be higher, or the premium amount would not be so important in lapse prediction, but in this particular case it could really lead to surrender *en masse*. Notice that this does not occur for $D[N, p(N)]$, because the usage of the probabilities smooths the results.
In particular, the impact on $TVOG[N, t^*(N)]$ is well explained by Figure 4.60. Even in the basic scenario with annual premium 1000, the average lapse year is around the tenth policy year, which is enough to reduce the average profit from about $D[ce, p(ce)] = 3500$ to about $D[N, t^*(N)] = 2000$. Even if the minimum guaranteed rate does not seem to be an important variable according to the Figure 4.35, it may be worth noting how it can impact the profit metrics. It is shown in Figure 4.38. Even at guaranteed rate 0%, the gap between $D[N, p(ce)]$ and $D[N, p(N)]$ as well as between $D[N, p(N)]$ and $D[N, t^*(N)]$ is quite large: it should be caused by the actual market conditions as well as other fixed tariff features. Indeed, all the three profit metrics decrease somewhat in parallel as the guaranteed rate increases, and this is also reflected in the TVOG (see Figure 4.39). Such a similar shape is due to the low importance of the minimum guaranteed rate, which cannot materially impact the lapse rate.
Given the regularity of the decrease in $D[N, t^*(N)]$, we expect a regular increase in average lapse year. Looking at Figure 4.61, it approximately grows in a linear fashion from about 8,5 years to about 11,5 years.
So far we focused our attention on tariff-related variables, which cannot change during the life of the contract. However, the crediting rate is especially a function of the fund performance. Given that `DeltaReturn` is

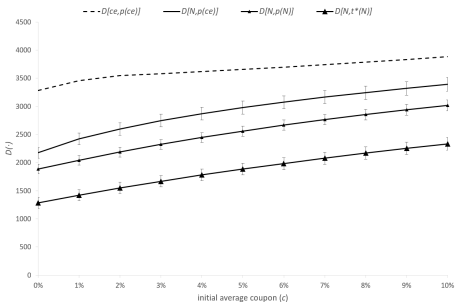


Fig. 4.40: Profit metrics by initial average coupon (FTSE MIB case)

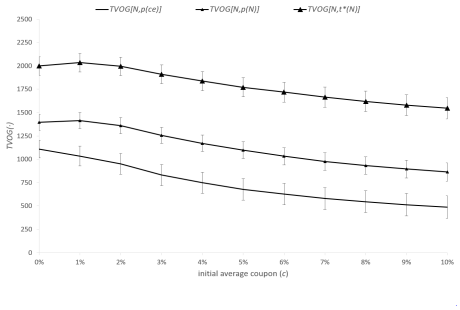


Fig. 4.41: TVOG metrics by initial average coupon (FTSE MIB case)

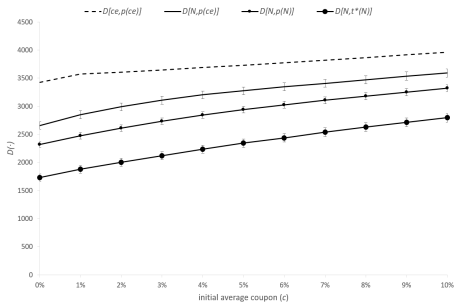


Fig. 4.42: Profit metrics by initial average coupon (EURO STOXX 50 case)

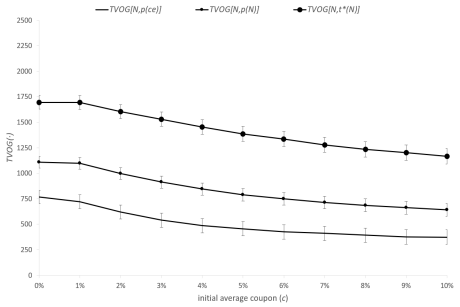


Fig. 4.43: TVOG metrics by initial average coupon (EURO STOXX 50 case)

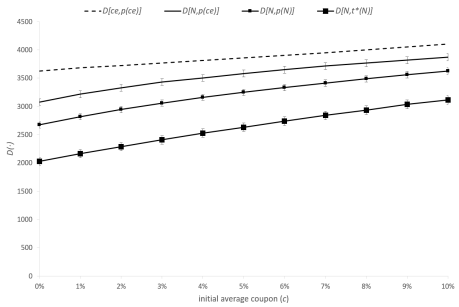


Fig. 4.44: Profit metrics by initial average coupon (S&P 500 case)

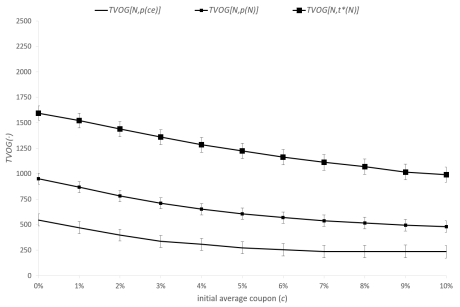


Fig. 4.45: TVOG metrics by initial average coupon (S&P 500 case)

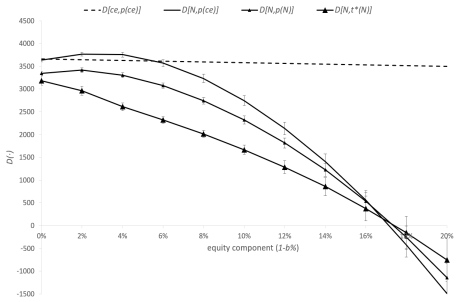


Fig. 4.46: Profit metrics by equity percentage (FTSE MIB case)

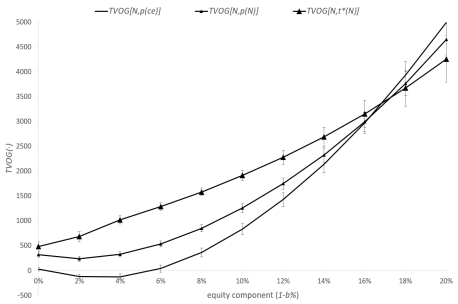


Fig. 4.47: TVOG metrics by equity percentage (FTSE MIB case)

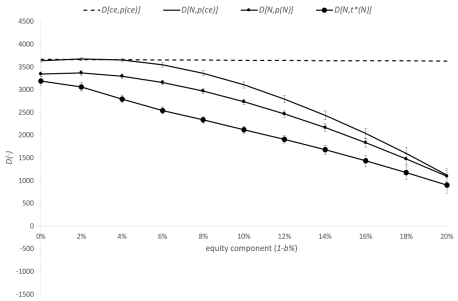


Fig. 4.48: Profit metrics by equity percentage (EURO STOXX 50 case)

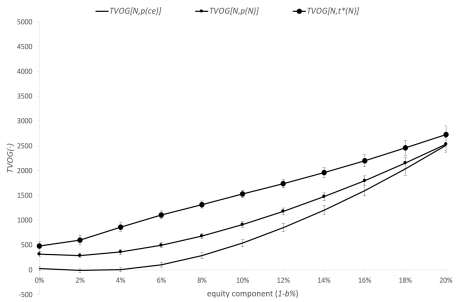


Fig. 4.49: TVOG metrics by equity percentage (EURO STOXX 50 case)

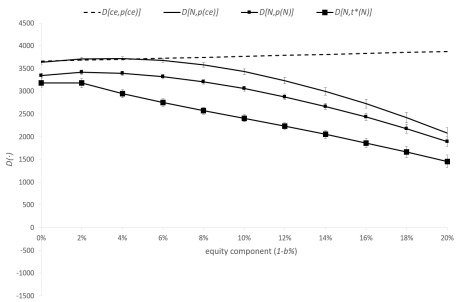


Fig. 4.50: Profit metrics by equity percentage (S&P 500 case)

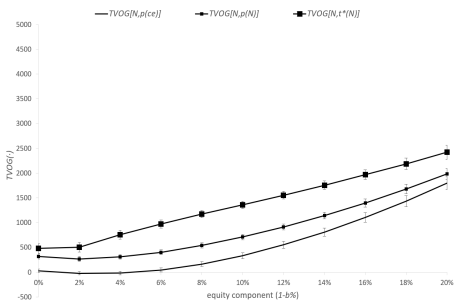


Fig. 4.51: TVOG metrics by equity percentage (S&P 500 case)

among the most important variables in predicting surrender activity (see Figure 4.35), we will consider variations in the fund-related parameters, in order to observe how the profit metrics and the lapse year are impacted by the market condition.

On the base of Figure 4.3, the initial average performance of the bond component is 3%, but now we let it change between 0% to 10% adjusting the coupon rates of the three BTPs by $\pm 1\%$. Notice that these parameters are used in the first policy years only, since new bonds will be bought as soon as the initial bonds mature. New bonds will return the new (stochastic) forward rate though. As a consequence, Figures 4.40-4.45 show an increase in profit and the related decrease in TVOG as the initial coupon rate increases. At the same time, $D[N, p(ce)]$, $D[N, p(N)]$ and $D[N, t^*(N)]$ get closer and closer to $D[ce, p(ce)]$, and this is especially evident in Figure 4.44. In this case, the minimum guaranteed rate has small impact on profit because of the higher returns guaranteed by S&P 500 (see Figure 4.4) as well as the growing initial coupon rate. As we can see in Figure 4.62, the higher the initial coupon rate, the later the policyholder's lapse activity, and this also contributes to the increase in $D[N, t^*(N)]$ towards $D[ce, p(ce)]$.

However, Figure 4.62 reveals some gap between the average lapse year among the three different equity investments. While the curves for EURO STOXX 50 and S&P 500 appear approximately parallel, the curve for FTSE MIB grows more irregularly. In fact, it shares the same average lapse year of the EURO STOXX 50 case when the coupon rate is zero, but the gap steadily increases proportionally to the coupon rate. Indeed, when the initial bond yield component is quite low, the policyholder lapses relatively early, as soon as he/she can find higher yields on the market. This especially occurs for low-return equity investments like FTSE MIB and EURO STOXX 50 (see Figure 4.4) since bond yields have the major impact on the fund performance.

The last set of parameters considered in our analysis involves the equity allocation percentage of the fund. Since Italian segregated funds may incorporate just a minor equity investment (and real estate, which we do not consider here), typically upper-bounded at 20%-30%, we assume $1 - b \in [0\%, 20\%]$. We also repeat the analysis for the three different equity indexes in Figure 4.4.

A first, strange effect we should explain is well evident in Figure 4.46 above all. When the equity percentage is low, $D[N, p(ce)]$ is slightly greater than $D[ce, p(ce)]$. Theoretically, this is not admissible, because $D[N, p(ce)]$ is affected by the rate guarantee in the disadvantageous scenarios, while $D[ce, p(ce)]$ is affected by the certainty equivalent scenario only. The direct consequence of this distortion is the negative TVOG in Figure 4.47 for the same low equity percentages. However, this anomalous gap is quite slight, probably due to the frequent negative-rate scenarios, which is a quirk of the Gaussian model, so we will not investigate it further.

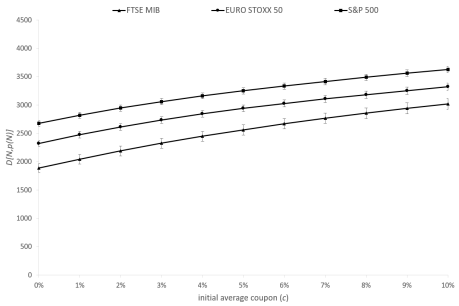


Fig. 4.52: $D[N, p(N)]$
by initial average coupon
and equity investment

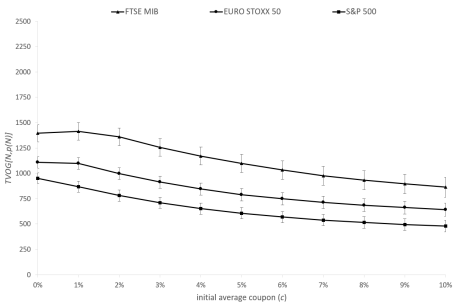


Fig. 4.53: $TVOG[N, p(N)]$
by initial average coupon
and equity investment

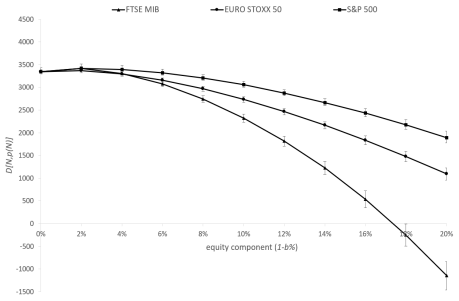


Fig. 4.54: $D[N, p(N)]$
by equity percentage
and equity investment

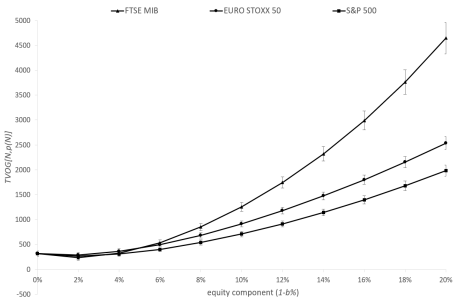


Fig. 4.55: $TVOG[N, p(N)]$
by equity percentage
and equity investment

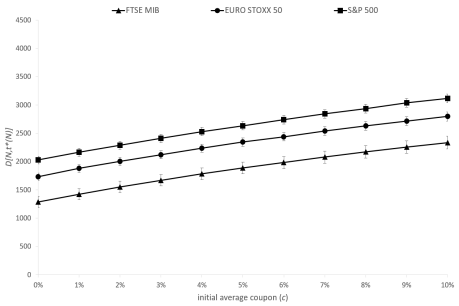


Fig. 4.56: $D[N, t^*(N)]$
by initial average coupon
and equity investment

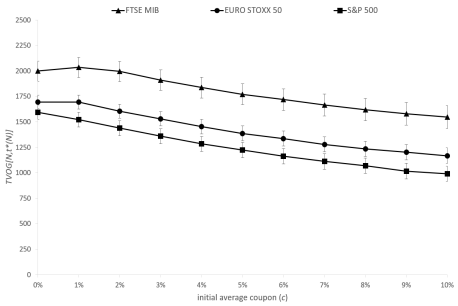


Fig. 4.57: $TVOG[N, t^*(N)]$
by initial average coupon
and equity investment

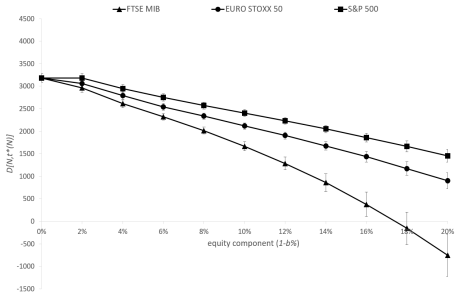


Fig. 4.58: $D[N, t^*(N)]$ by equity percentage and equity investment

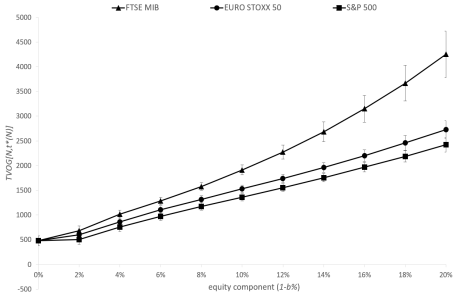


Fig. 4.59: $TVOG[N, t^*(N)]$ by equity percentage and equity investment

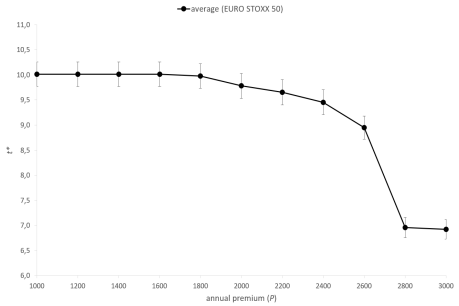


Fig. 4.60: Average lapse year by annual fair premium (EURO STOXX 50 case)

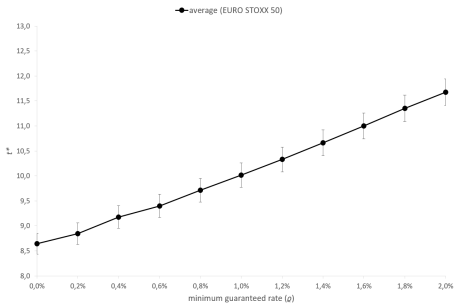


Fig. 4.61: Average lapse year by minimum guaranteed rate (EURO STOXX 50 case)

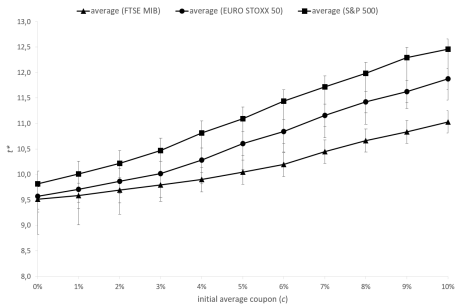


Fig. 4.62: Average lapse year by initial average coupon and equity investment

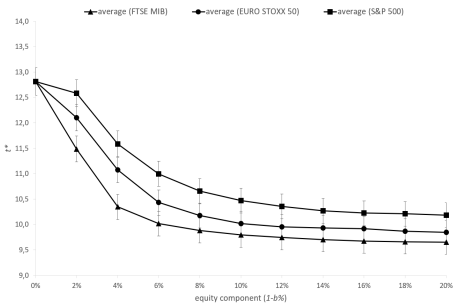


Fig. 4.63: Average lapse year by equity percentage and equity investment

Without any equity component, profits (see Figure 4.54 and Figure 4.58) and TVOGs (see Figure 4.55 and Figure 4.59) are obviously equal, slightly above 3000 and around 500 respectively. Notice that it is in line with the total loading from the contract, that is, $1000 \times 15\% \times 20 = 3000$, since the crediting rate matches the deflator whereas no investment profit comes from any equity component. Average profit strongly decreases as the equity allocation grows regardless of the market index and the profit metric. The main reason is well represented by the Figure 4.63: whereas the policyholder tends to lapse later during the life of the contract if the equity component is residual, when it is approximately above 5% the average lapse year stabilizes at about 10-10,5 years (depending on the market index) since he/she starts profiting from high-yield equity scenarios, while being protected by the guarantee in low-yield scenarios. Such a persistent behaviour seems to be worth more than the lapse itself if the equity component is, indeed, material. Naturally, the policyholder's persistence in downward scenarios means a more and more significant loss for the company.
Reasonably, the only average loss occurs when investing in FTSE MIB (see Figure 4.46 as well as Figures 4.54 and 4.58), which yields an average negative return as reported in Figure 4.4. Such a return allows for a further phenomenon, which is less evident with more profitable equity investments. Figure 4.46 shows that $D[N, p(ce)]$ $D[N, p(N)]$ could be even smaller than $D[N, t^*(N)]$ for relatively high equity percentages - higher than 16%. This may sound counter-intuitive, but the reason is quite straightforward. If the portfolio is affected by the negative returns of some assets, it will be sustainable as long as the losses are absorbed by the loadings. If it is not the case, the company will realize losses until maturity potentially. Paradoxically, it would benefit from the eventual lapse of the policyholder, because it would avoid the losses of the remaining policy years. All in all, the full lapse at the end of the year t^* is beneficial as compared to the probability-based lapse, which keeps the contract in force until maturity. This effect is also reflected on the TVOG in Figure 4.47, showing that it gets lower in the time-to-lapse approach by high equity percentages.

4.5.4 TVOG decomposition

A last result to mention regards the relative impact that an active PHB may have on the TVOG of an insurance contract. In effect, it may be negligible in some situations, whereas it could require some pricing adjustment in others.
It is worth clarifying that the TVOG definition implicitly considered so far is a *real-world* definition, because the return of the assets is always adjusted by a spread (represented by the parameters d_x and d_y for the bonds) or a risk premium (i.e., μ_S for the equities). Of course, this is not in line with risk-neutral frameworks such as Solvency II, but the concept of real-

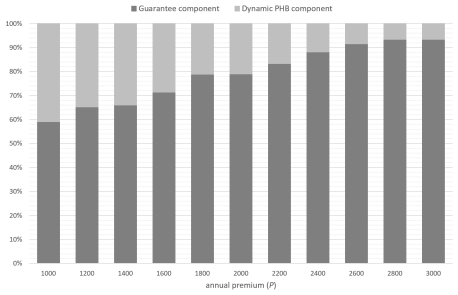


Fig. 4.64: $TVOG[N, p(N)]$ guarantee-PHB split by annual fair premium (EURO STOXX 50 case)

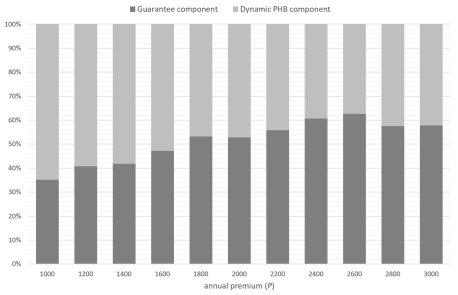


Fig. 4.65: $TVOG[N, t^*(N)]$ guarantee-PHB split by annual fair premium (EURO STOXX 50 case)

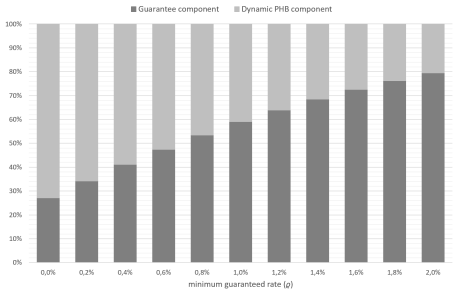


Fig. 4.66: $TVOG[N, p(N)]$ guarantee-PHB split by minimum guaranteed rate (EURO STOXX 50 case)

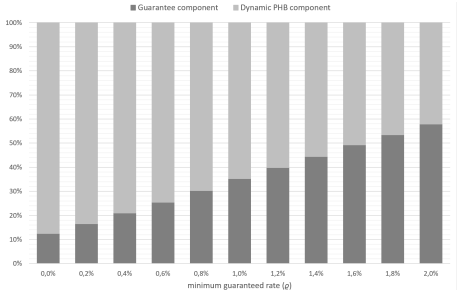


Fig. 4.67: $TVOG[N, t^*(N)]$ guarantee-PHB split by minimum guaranteed rate (EURO STOXX 50 case)

world TVOG is rather significant as long as one uses it to assess real-world performances of single products (for instance, it may be useful to calibrate surrender penalties in pricing). However, take into account that a real-world TVOG is theoretically lower than a risk-neutral TVOG, because spreads and risk premiums should contribute to higher profits.

It is worth recalling that $TVOG[N, p(ce)]$ includes the impact of the minimum guaranteed rate, but not that related to dynamic PHB. On the contrary, $TVOG[N, p(N)]$ and $TVOG[N, t^*(N)]$ include both the effects. This is basically the reason why we can assume that $D[N, p(N)]$ and $D[N, t^*(N)]$ are smaller than $D[N, p(ce)]$, just like the various plots in Subsection 4.5.3 have shown.

The gap between $TVOG[N, p(ce)]$ and both $TVOG[N, p(N)]$ and $TVOG(t^*)$ is explained by PHB only, and it represents a measure for the surrender op-

tion's value, say V_{PHB}. In formulas, we can write

$$TVOG[N, p(N)] = TVOG[N, p(ce)] + V_{PHB}(p) \quad (4.64)$$
$$TVOG[N, t^*(N)] = TVOG[N, p(ce)] + V_{PHB}(t^*) \quad (4.65)$$

and it is interesting to study the impact of each component across different scenarios and parametrizations as a percentage of $TVOG[N, p(N)]$ and $TVOG[N, t^*(N)]$ respectively:

$$1 = \frac{TVOG[N, p(ce)]}{TVOG[N, p(N)]} + \frac{V_{PHB}(p)}{TVOG[N, p(N)]} \quad (4.66)$$
$$1 = \frac{TVOG[N, p(ce)]}{TVOG[N, t^*(N)]} + \frac{V_{PHB}(t^*)}{TVOG[N, t^*(N)]}. \quad (4.67)$$

This is represented in Figures 4.64-4.78 for each of the parametrization we have already considered in Subsection 4.5.3.
First, the PHB component gets smaller and smaller as the annual premium increases (see Figures 4.64-4.65). Given that the premium amount is a rather important variable in lapse prediction (see Figure 4.35), that may sound counter-intuitive. However, higher premiums lead to higher guaranteed sum assured. All in all, the absolute difference in TVOG is rather constant regardless of the premium amount, while the certainty equivalent profit increases much more steeply. This effect implies greater guarantee components by higher premiums. Nonetheless, if premiums are low enough, the PHB component may be greater than 40% for $TVOG[N, p(N)]$ and even 60% for $TVOG[N, t^*(N)]$.
Similarly, the PHB component decreases linearly as the minimum guaranteed rate increases (see Figures 4.66-4.67). With minimum guaranteed rate 0%, the PHB impact is higher than 70% for $TVOG[N, p(N)]$ and almost 90% for $TVOG[N, t^*(N)]$, which is not surprising since the guarantee barely plays a role in the TVOG determination. Nonetheless, the greater the guaranteed rate, the more significant its impact on TVOG, the less significant the PHB impact on TVOG. At high guarantees, the policyholder has much fewer reasons to lapse, and the TVOG will be entirely function of the guarantee itself. In our analysis, however, the smallest PHB impact on the TVOG is observed when the minimum guaranteed rate is equal to 2%. In Figures 4.66 and 4.67, the TVOG is split between a guarantee rate component around 80% and 60% against a PHB component around 20% and 40%.
Figures 4.68, 4.70 and 4.72 as well as Figures 4.69, 4.71 and 4.73 look relatively similar. As the initial average coupon from the bonds in the segregated fund increases, the PHB impact slightly grows regardless of the equity investment. In particular, the smallest PHB components appear by the riskiest indices and the lowest coupon rates. For instance, Figure 4.68

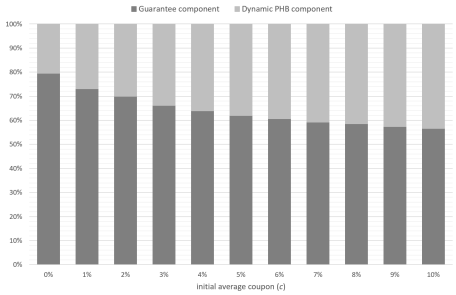


Fig. 4.68: $TVOG[N, p(N)]$ guarantee-PHB split by initial average coupon (FTSE MIB case)

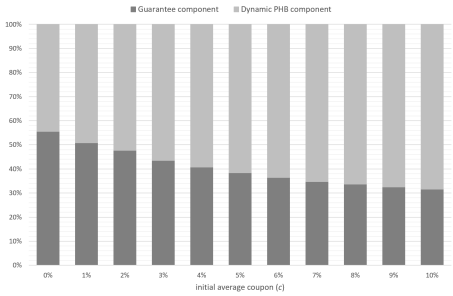


Fig. 4.69: $TVOG[N, t^*(N)]$ guarantee-PHB split by initial average coupon (FTSE MIB case)

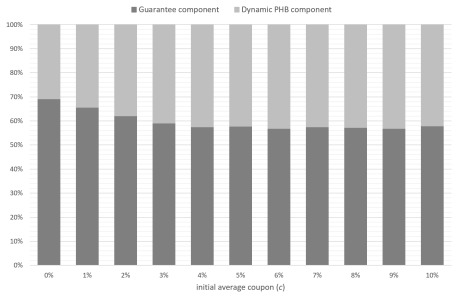


Fig. 4.70: $TVOG[N, p(N)]$ guarantee-PHB split by initial average coupon (EURO STOXX 50 case)

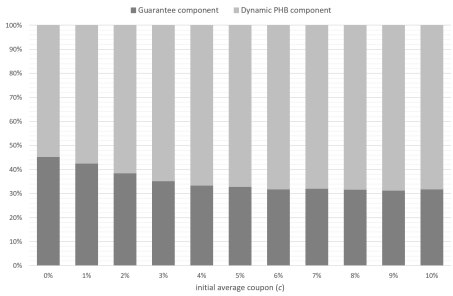


Fig. 4.71: $TVOG[N, t^*(N)]$ guarantee-PHB split by initial average coupon (EURO STOXX 50 case)

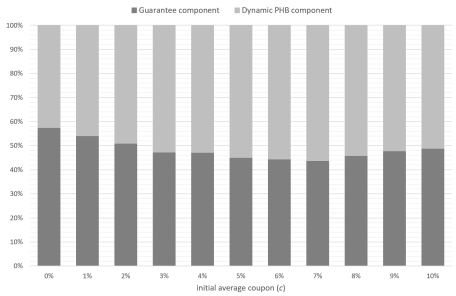


Fig. 4.72: $TVOG[N, p(N)]$ guarantee-PHB split by initial average coupon (S&P 500 case)

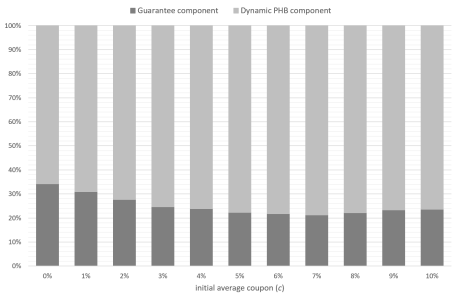


Fig. 4.73: $TVOG[N, t^*(N)]$ guarantee-PHB split by initial average coupon (S&P 500 case)

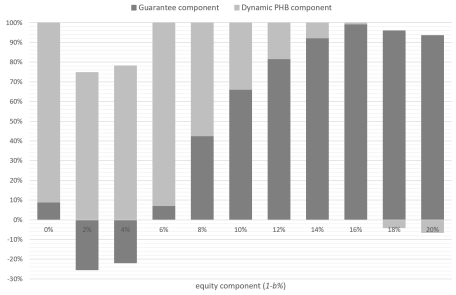


Fig. 4.74: $TVOG[N, p(N)]$ guarantee-PHB split by equity percentage (FTSE MIB case)

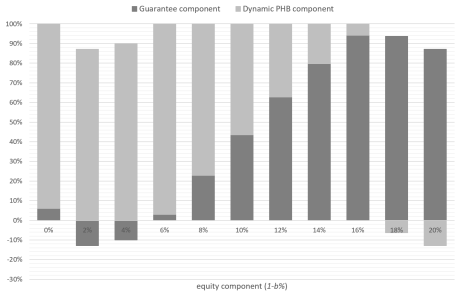


Fig. 4.75: $TVOG[N, t^*(N)]$ guarantee-PHB split by equity percentage (FTSE MIB case)

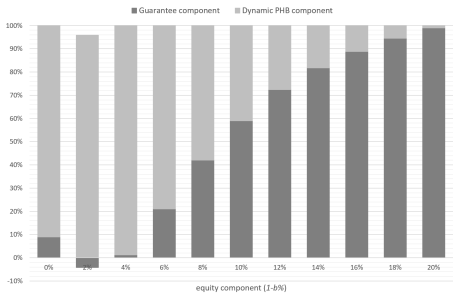


Fig. 4.76: $TVOG[N, p(N)]$ guarantee-PHB split by equity percentage (EURO STOXX 50 case)

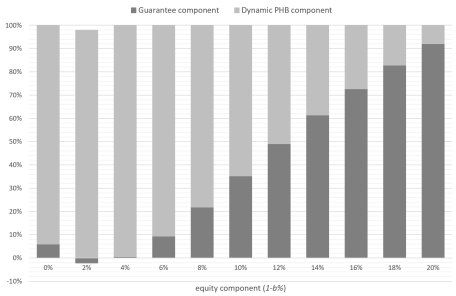


Fig. 4.77: $TVOG[N, t^*(N)]$ guarantee-PHB split by equity percentage (EURO STOXX 50 case)

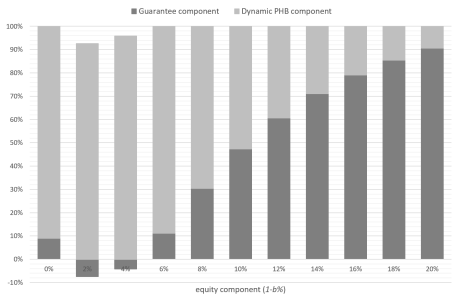


Fig. 4.78: $TVOG[N, p(N)]$ guarantee-PHB split by equity percentage (S&P 500 case)

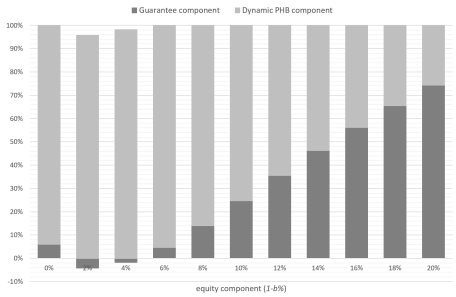


Fig. 4.79: $TVOG[N, t^*(N)]$ guarantee-PHB split by equity percentage (S&P 500 case)

reports a PHB component around 20% by coupon rate 0%. This is reasonable, because, from the perspective of the policyholder, it represents the best condition to profit of the guarantee option embedded in the contract. On the other hand, the PHB component can reach and exceed 50% when the equity indices is less risky and the coupon rate is high enough such as in Figure 4.72. In the time-to-lapse approach (see Figures 4.69, 4.71 and 4.73), the PHB component is even more significant, and regularly overcomes the guarantee component, reaching 80% in Figure 4.73.
In general, the reduction effect in the guarantee component is more evident for lower coupon rates. As the coupon rate increases, the split between the two components tends to stabilize at some level. Referring to the PHB component, such a level is around 40-50% and 70-80% in the probability-based approach and time-to-lapse-based approach respectively.
Figures 4.74, 4.76 and 4.78 as well as Figures 4.75, 4.77 and 4.79 show similar shapes as well. When the equity component is residual, the TVOG comes from PHB only, but remember we used an initial average coupon of 3% against a minimum guarantee rate of only 1%, that is, there is no impact from the guarantee in the first policy years (almost) regardless of the economic scenario. It can even lead to a negative TVOG for the guarantee component, for instance, by an equity component of 2-4%. Then, the introduction of a significant equity component increases the investment risk of the fund, thus the value of the guarantee option, and slows surrender activity down. In the hypothetical case of 100% investment in equity, the policyholder has no motivation to lapse: while he/she benefits from any upside in any equity scenario - by definition, more favourable than the correspondent government bond scenario - he/she feels protected by the guarantee in any downside scenario. Nonetheless, we don't need 100% equity to observe that effect: Figures 4.74, 4.76 and 4.78 shows guarantee components around 100% at a level of 20% invested in equity.
All in all, except for some specific situations such as high premium amounts (see Figure 4.64) or high equity percentages (see Figures 4.74, 4.76 and 4.78), the PHB component is always quite relevant and sometimes even greater than the guarantee component. For instance, it can be the only TVOG component by very low equity percentages (see Figures 4.74, 4.76 and 4.78). In such cases, neglecting dynamic PHB is practically meaning that our TVOG estimation is zero or very close to zero. This cannot be an acceptable simplification in light of our results. In fact, the PHB may be a significant component of the TVOG in a number of scenarios, conditions and parametrizations.

4.6 Limitations, extensions and conclusions

Although the analysis of this chapter is based on a specific, proprietary dataset, it provides us with a realistic idea of the relative impact of dynamic PHB. According to the CRISP-DM standard for the data mining process (see Section 1.2), the chapter can be formally broken as follows:

C1 *Business understanding*: Section 4.1 and 4.2

C2 *Data understanding*: Subsection 4.5.1

C3 *Data preparation*: Subsection 4.5.1

C4 *Modelling*: Section 4.3 and Subsection 4.5.2

C5 *Evaluation*: Subsection 4.5.3 and 4.5.4

C6 *Deployment*: none.

More specifically, in Section 4.3 we referred to a simplified ALM model, the same as in Aleandri (2016). Without a doubt, a real ALM model would be much more accurate and somewhat less questionable, because it would allow (at least theoretically) for a correct calculation of the segregated fund return as the ratio of revenue to average reserve. Actually, our model implicitly assumes that the market-value-based asset total return coincides with the balance-sheet-based segregated fund return: although it is a good proxy, they are not equivalent.
In particular, we do not account for available-for-sale securities in portfolio, that is, all those securities (whether bonds or stocks) that the company may trade at its own discretion to realize gains or losses. In fact, we considered held-to-maturity securities only, although insurance companies generally hold available-for-sale securities as well. On the other hand, segregated fund allocations tend to be quite stable among insurance companies, in order to guarantee stable returns. This is the reason why we can accept our simplifications, avoiding a number of complications that would be beyond the scope of the research.
Even if we recognize that some important variables have not been included among the predictors of the PHB (e.g., unemployment rate and policyholder's salary), our results are globally in line with those of other similar studies. Our analysis has brought out some typical risky profiles and rational behaviours. Positive correlation between age and lapse tendency is confirmed: oldest people surrender more than younger people. At the same time, higher premiums make policyholders more prone to lapse, which is quite plausible. The performance of the contract plays an important role as well. Indeed, when the contract cannot return a yield comparable to the actual market yields, lapse is more likely. To some extent, it proves the presence of rational behaviour of the policyholders since higher return

discrepancies can be either due to poor segregated fund returns (leading to an arbitrage-oriented behaviour) or higher bond spreads (leading to a crisis-oriented behaviour).
The Section 4.5.4 completed the analysis by focusing on the effective impact of dynamic PHB modelling on the TVOG. In particular, all the histograms in Figures 4.64-4.79 reveal how significant the unique behaviour of a policyholder can be on the profit valuation. Most of the time, indeed, the TVOG due to PHB covers a significant portion of the total TVOG. In other terms, the TVOG calculated without dynamic PHB assumptions might materially underestimate the total TVOG. We should be aware of the fact that it is due to current economic conditions as much as intrinsic features of the specific policyholder. Both of them should be taken into account for a comprehensive and prudential profit valuation.

4.6.1 Key conclusions for actuarial practitioners

This last application firstly aimed to show the potential instability of many machine learning techniques such as decision trees. When using them, actuaries should always remember that any model other than regression does not guarantee variance minimization. In practice, this means that overfitting is a major issue in data science, and needs to be properly addressed. By contrast, regression models suffers from the opposite problem, that is, underfitting: while they are structured to minimize variance, they can hardly reduce bias.
Nonetheless, this chapter is also aimed to provide a solution to the problem of overfitting, beyond the usual tuning of meta-parameters. In data science, ensembles are often preferred over standalone techniques precisely because they guarantee a higher level of stability, maintaining the advantage in terms of flexibility and accuracy. Indeed, we demonstrated that bagging trees materially reduced overfitting in lapse rate estimation as compared to a single classification tree.
However, actuaries should take into account the considerable loss in interpretability when using ensembles. As we pointed out in this chapter, single trees offer an intuitive, tree-shaped illustration of relationships among variables, while this is no longer available as soon as a number of different trees are combined (regardless of the combination rule). Even though there is a range of tools to improve result interpretation (e.g., variable importance), the black-box effect may be unacceptable, even in the presence of a relevant increase in accuracy. This is especially true in an industry like insurance, where result communication is often crucial.

Chapter 5

Final Thoughts

In the light of the current technological evolution in finance and insurance, actuaries need to embrace topics and methods which are seldom part of their toolkit. Even if they are popping up now thanks to the major advancements in data availability and computational power, they are based on ideas developed many years ago in academia. At that time, however, practical reasons prevent their applicability in industry, which has always preferred simplicity over accuracy. This is true in insurance and actuarial practice as well. In spite of the significant progresses in the last decade, there is still room for improvement. This work was indeed meant to provide a practical introduction to the most common topics in data science with an eye to potential enhancements in the insurance sector.

In Chapter 2, we described some ways to use unsupervised learning for data preparation as a preliminary step before supervised learning. For instance, cluster analysis helped us to detect the most informative groups of records, while association rules determined which features could most likely lead to rider purchase. Nonetheless, the main results of that chapter relates to the renewal rate prediction in third-party liability insurance and the detection of the features that affect it.

Chapter 3 demonstrated how alternative machine learning techniques such as decision trees can improve the accuracy of a classical actuarial exercise: non-life claim reserving. At the same time, it showed as well that this is not always the case. Indeed, while decision trees could outperform regression models in claim payment amount prediction, no improvement was possible for closing delay estimation. Indeed, as opposed to GLMs, those alternative techniques do not provide any theoretical guarantee of "minimal variance" or "best fitting", but their flexibility can sometimes lead to better performance.

Using the same techniques to exploit such a gain in accuracy for lapse rate estimation in Chapter 4 too, we instead faced their main pitfall: instability. This gave an argument to introduce the concept of ensemble as a possible

solution. In particular, bagging trees not only largely reduced the original overfitting, but it even outperformed regression. As a consequence, its lapse rate estimates were used for stochastic profit evaluation of a life product. Setting several parametrizations, we observed the impact of dynamic policyholder behaviour on the profit in terms of TVOG, separating it from the impact due to the minimum guaranteed rate.

All in all, the main goal of this dissertation is to provide actuaries with a practical guide on data science, conveying the message that it can be useful in the daily practice. Reading it, actuarial practitioners from any background should be able to understand the main data science concepts, the data-driven approach based on dataset partitioning, the most common unsupervised and supervised algorithms as well as their technicalities and, in general, many useful details for the practice. More importantly, they can figure out various ways to use data science as an enhancement of their daily activity to tackle traditional problems. These include probability estimation for actuarial assumptions (as we did in Chapter 2), claim amount prediction for pricing as well as reserving (as we did in Chapter 3), risk-based capital model refinement (as we did in Chapter 4) and so on. The case studies have been structured to highlight advantages and disadvantages of data-driven approaches. They should especially help actuaries to realize that data science is neither better nor worse than traditional statistics. Instead, it is an alternative to leverage available information and achieve higher accuracy or new insights at the cost of potential instability and increased model complexity.

Nevertheless, this work was not meant to be fully comprehensive. Many other topics could have been included on the data science side as well as the actuarial side. Subsection 1.6.2 has already mentioned a number of out-of-scope methods such as regularized linear models and support vector machines. Additionally, a number of more recent topics are catching on in data science, and researchers are already trying to exploit them in insurance. They include unstructured data handling, natural language processing, deep learning, image recognition and many others. For instance, claim management cost in motor insurance could be largely reduced, if an algorithm were able to assess the damage by parsing a simple picture of the damaged vehicle. Another, tricky aspect of claim management is fraud detection. First of all, fraudsters are extremely few (less than 1% according to IABE Information Paper (2015)), and, more importantly, not all of them succeed, so unsuccessful attempts will not be recorded as frauds, and potential fraudulent customer will not be recorded as fraudulent customers. The major concern is indeed the detection of potential fraudulent customers, who are likely to defraud in the future. These are some of the ideas representing a different, higher level of research applied to insurance business, involving unconventional data as well as unconventional techniques.

More specific actuarial research is being impacted as well. Some papers

on actuarial case studies tackled with data science (many of them cited throughout this dissertation) have already been published in the last few years, while quality and quantity will quickly increase in the near future. Without a doubt, non-life pricing represents one of the most successful actuarial applications, and data science is permanently part of the process in several non-life companies. Nonetheless, many other topics are coming out little by little: beside customer management, individual reserving and policyholder behaviour, we should also mention interest rate modelling (see Puri (2016)), mortality estimation (see Levantesi et al. (2019)) and capital valuations (see Castellani et al. (2019)), among others. Future applications may include strategic asset allocation, strategic discretionary profit sharing and many others: the opportunities are potentially limitless, as long as proper data is available.
In fact, the insurance landscape is changing, and so the actuarial profession, but this should be intended as an enhancement of the typical actuarial profile rather than a mere shift of skills. Data science should complement traditional actuarial topics rather than replace them, because it is not only about a set of algorithms and methods, but it is an entirely new approach to solve problems. The emphasis throughout these pages on concepts like data manipulation, training and validation, accuracy and variance, does not relate to any specific algorithm, but a broad data-driven perspective, which turns out to be necessary dealing with big data above all. As explained in IFoA (2018), such an enhancement will be primarily visible in six aspects of the daily actuarial practice:

- *data quality*: as soon as data-driven techniques are widely proven to perform better than traditional ones, data quality will improve as a consequence, leading to greater comfort in actuarial modelling, whether traditional or non-traditional;

- *data sources*: although actuaries are used to work on plain, numerical datasets, machine learning potentially opens up opportunities for actuaries to explore alternative data sources, pushing them out of their comfort zone, and possibly leading them to new insights;

- *modelling techniques*: as shown throughout this dissertation, data-driven techniques - whether standard or customized - can outperform traditional techniques in terms of accuracy, and, to some extent, lead to new insights and drive business decisions;

- *problem solving*: a greater variety of techniques and methods can lead to new approaches to solve common problems, for instance, unsupervised learning, backtesting with data science tools, machine-learning-based data manipulation and so on;

- *analysis speed*: when data quality is high and tools are automatized, the whole process gets agile enough to avoid huge effort on cleaning data and launching calculations, so that actuaries can invest much more time in productive activities such as result interpretation;
- *data visualization*: reporting and communication to non-actuarial audience, one of the most important skills for actuaries, can materially improve by using new data visualization techniques.

Another, less immediate benefit to actuaries does not relate to their current daily practice. Rather, it relates to the possibility of expanding the profession into new areas in the future. Actuaries have the unique chance to exploit the huge amount of opportunities offered by data science. The insurance industry is improving in that direction, and they should lead the change, accepting new challenges rather than merely relying on tradition.

Acknowledgements The author wishes to thank his promoters and supervisors Prof. Susanna Levantesi (Università "La Sapienza", Rome) and Dr Joseph Lo (Institute and Faculty of Actuaries, London) for their continuous support and precious feedbacks since the very first draft of this dissertation. Also, special thanks go to Emanuele Marchi (Università "La Sapienza", Rome) for his substantial contribution to Chapter 2. Finally, the author is grateful to his SA0 examiner, Pietro Parodi (Institute and Faculty of Actuaries, London), as well as his two doctoral examiners, Prof. Marco Pirra (Università della Calabria, Cosenza) and Prof. Gian Paolo Clemente (Università Cattolica del Sacro Cuore, Milano), for accepting to review the dissertation.

Bibliography

R. Agrawal, T. Imielinski, A. Swami, *Mining association rules between sets of items in large databases*, Proceedings of the 1993 ACM SIGMOD international conference on Management of data, 1993.

H. Akaike, *Information Theory and an Extension of the Maximum Likelihood Principle*, Proceedings of the 2nd International Symposium on Information Theory, 1973.

M. Aleandri, *Valutazione del rischio di mercato di prodotti tradizionali in gestione separata in base al nuovo regime informativo per i PRIIPs*, Dept. of Statistical Sciences at Univ. "La Sapienza", Rapporto Tecnico n. 7, 2016.

M. Ankerst, M. M. Breunig, H. Kriegel, J. Sander, *OPTICS: Ordering Points To Identify the Clustering Structure*, ACM SIGMOD International Conference on Management of Data, ACM Press, pp. 49-60, 1999.

K. Antonio, E. Godecharle, R. Van Oirbeek, *A Multi-State Approach and Flexible Payment Distributions for Micro-Level Reserving in General Insurance*, SSRN Manuscript ID 2777467, 2016.

K. Antonio, R. Plat, *Micro-level stochastic loss reserving for general insurance*, Scandinavian Actuarial Journal, Vol. 7, pp. 649-669, 2014.

K. Arai, A. R. Barakbah, *Hierarchical K-means: an algorithm for centroids initialization for K-means*, Reports of the Faculty of Science and Engineering, Vol. 36, pp. 25-31, 2007.

B. Avanzi, B. Wong, X. Yang, *A Micro-Level Claim Count Model with Overdispersion and Reporting Delays*, SSRN Manuscript ID 2705241, 2016.

A. Azevedo, M. F. Santos, *KDD, SEMMA and CRISP-DM: a parallel overview*, Proceedings of the IADIS European Conference on Data Mining, 2008.

D. F. Babbel, *Asset-Liability Matching in the Life Insurance Industry* in E. Altman and I. Vanderhoof (eds.), *The Financial Dynamics of the Insurance Industry*, Irwin Professional Publishing, 1995.

A. L. Badescu, L. X. Sheldon, D. Tang, *A Marked Cox Model for IBNR Claims: Model and Theory*, Insurance: Mathematics and Economics, Vol. 69, pp. 29-37, 2016.

M. J. A. Berry, G. S. Linoff, *Data Mining Techniques*, Wiley, 1997.

P. Billingsley, *Probability and Measure*, Wiley, 1995.

R. L. Bornhuetter, R. E. Ferguson, *The actuary and IBNR*, Proceedings of the Casualty Actuarial Society, Vol. 59, pp. 181-195, 1972.

L. Breiman, *Bagging predictors*, Machine Learning, Vol. 24/2, pp. 123-140, 1996.

L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, *Classification and Regression Trees*, Wadsworth Statistics/Probability Series, 1984.

D. Brigo, F. Mercurio, *Interest Rate Models: Theory and Practice*, Springer, 2001.

J. Brownlee, *A Tour of Machine Learning Algorithms*, Machine Learning Mastery, Vol. 25, 2013.

G. E. Cannon, *A Study of Persistency*, RAIA XXXVII, 1948.

G. Castellani, U. Fiore, Z. Marino, L. Passalacqua, F. Perla, S. Scognamiglio, P. Zanetti, *An Investigation of Machine Learning Approaches in the Solvency II Valuation Framework*, SSRN Manuscript ID 3303296, 2019.

A. L. Cauchy, *Sur l'équation à l'aide de laquelle on détermine les inégalités séculaires des mouvements de planèts*, Oeuvres Complètes (IIème Série), Vol. 9, Blanchard, 1829.

R. R. Cerchiara, M. Edwards, A. Gambini, *Generalized Linear Models in Life Insurance: Decrements and Risk factor analysis under Solvency II*, Giornale dell'Istituto Italiano degli Attuari, 2009.

P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, R. Wirth, *CRISP-DM 1.0 Step-by-step data mining guide*, pp. 10-12, 2000.

A. Charpentier, *Big Data and Machine Learning with an Actuarial Perspective*, Institut des Actuaires en Belgique (IABE) Summer School, Louvain-la-Neuve, 2015.

A. Charpentier, *Computational Actuarial Science with R*, CRC Press, pp. 499-501, 2015.

A. Charpentier, C. Villa, *Generating Yield Curve Stress-Scenarios*, HAL, pp. 1-44, 2010.

H. Chen, R. Chen, T. Chen, Y. Chen, S. Li, C. Lin, T. Tsai, *A combined K-means and hierarchical clustering method for improving the clustering efficiency of microarray*, International Symposium on Intelligent Signal Processing and Communication Systems, pp. 405-408, 2005.

W. Chen, W. Kuo, C. Tsai, *An Empirical Study of the Lapse Rate: The Cointegration Approach*, Journal of Risk and Insurance, 2003.

J. G. R. Crombie, K. G. Forman, P. R. Gibbens, D. C. Mason, M. D. Paterson, P. C. Shaw, J. M. G. Smart, H. Smith, C. G. Thomson, R. G. Thomson, *An Investigation into the Withdrawal Experience of Ordinary Life Business*, Transactions of Faculty of Actuaries, Vol. 36, 1979.

A. Dar, C. Dodds, *Interest Rates, the Emergency Fund Hypothesis and Saving through Endowment Policies: Some Empirical Evidence for the UK*, Journal of Risk and Insurance, 1989.

P. De Jong, G. Z. Heller, *Generalized Linear Models for Insurance Data*, Cambridge University Press, 2008.

P. Deprez, P. V. Shevchenko, M. V. Wüthrich, *Machine Learning Techniques for Mortality Modeling*, SSRN Manuscript ID 2921841, 2017.

C. Dutang, A. Charpentier, *Package 'CASdatasets'*, pp. 4-5, 2016.

B. Efron, *Bootstrap methods: another look at the jackknife*, Annals of Statistics, Vol. 7/1, pp. 1-26, 1979.

European Insurance and Occupational Pensions Authority (EIOPA), *Technical Specification for the Preparatory Phase (Part 1)*, pp. 74 and 211, 2014.

European Insurance and Occupational Pensions Authority (EIOPA), *The underlying assumptions in the standard formula for the Solvency Capital Requirement calculation*, pp. 14-15, 2014.

M. Ester, H. Kriegel, J. Sander, X. Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press, pp. 226-231, 1996.

V. Estivill-Castro, *Why so many clustering algorithms - A Position Paper*, ACM SIGKDD Explorations Newsletter, Vol. 4/1, pp. 65–75, 2002.

European Commission, *Annexes to the Supplementing Regulation (EU) No 1286/2014 of the European Parliament and of the Council on key information documents for packaged retail and insurance-based investment products (PRIIPs)*, 2016.

U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, *From Data Mining to Knowledge Discovery in Databases*, AI Magazine, Vol. 17/3, 1996.

U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, *Knowledge Discovery and Data Mining: Towards a Unifying Framework*, KDD-96 Proceedings, 1996.

U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy *Advances in Knowledge Discovery and Data Mining*, MIT Press, 1996.

E. W. Frees, R. A. Derrig, G. Meyers, *Predictive Modeling Applications in Actuarial Science. Volume I: Predictive Modeling Techniques*, Cambridge, 2014.

E. W. Frees, R. A. Derrig, G. Meyers, *Predictive Modeling Applications in Actuarial Science. Volume II: Case Studies in Insurance*, Cambridge, pp. 159-179, 2016.

Y. Freund, R. E. Schapire, *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*, Journal of Computer and System Sciences, Vol. 55/1, pp. 119-139, 1997.

P. Geurts, D. Ernst, L. Wehenkel, *Extremely randomized trees*, Machine Learning, Vol. 63, pp. 3-42, 2006.

L. Guelman, M. Guillen, A. M. Pérez-Marín, *A decision support framework to implement optimal personalized marketing interventions*, Decision Support Systems, Vol. 72, pp. 24-32, 2015.

D. Gulrajani, B. P. Murphy, D. Won, *Valuation and risk analysis of international bonds*, The handbook of fixed income securities, Irwin, 1995.

L. Guo, *Applying data mining techniques in property/casualty insurance*, Casualty Actuarial Society Forum, pp. 1-25, 2003.

T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*, Springer Series in Statistics, 2009.

M. Hiabu, C. Margraff, M. D. Martinez-Miranda, J. P. Nielsen, *The link between classical reserving and granular reserving through double chain ladder and its extensions*, British Actuarial Journal, Vol. 21/1, pp. 97-116, 2016.

T. K. Ho, *Random Decision Forests*, Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14-16 August 1995, pp. 278-282, 1995.

K. Hornik, M. Stinchcombe, H. White, *Multilayer Feedforward Networks are Universal Approximators*, Neural Networks, Vol. 2, pp. 359-366, 1989.

H. Hotelling, *Analysis of a complex of statistical variables into principal components*, J Educ Psychol, Vol. 25, pp. 417-441, 1933.

R. E. Hoyt, *Modeling Insurance Cash Flows for Universal Life Policies*, Journal of Actuarial Practice, 1994.

Y. Hwang, P. Lu, *Empirical Analysis of Surrender in the Taiwan Life Insurance Companies*, http://www.wriec.net/wp-content/uploads/2015/07/5B2_Hwang.pdf, 2014.

Institut des Actuaires en Belgique (IABE), *BIG DATA: An actuarial perspective*, Information Paper IABE, pp. 1-23, 2015.

International Accounting Standards Board (IASB), *IFRS 17 Insurance Contracts*, pp. 56, 2017.

Institute and Faculty of Actuaries (IFoA), *Practical Application of Machine Learning Within Actuarial Work*, 2018.

Istituto Nazionale di Statistica (ISTAT), *Tavole di mortalità*, 2015.

S. Jamal, *Non-Life methodologies applied to lapse rate modeling*, Astin Colloquium, 2016.

G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning. With Applications in R*, Springer Texts in Statistics, 2015.

F. Jamshidian, Y. Zhu, *Scenario Simulation: Theory and Methodology*, Finance and Stochastics, pp. 43-67, 1997.

P. J. Jennings, *Using Cluster Analysis to Define Geographical Rating Territories*, Discussion Paper Program, Casualty Actuarial Society, pp. 34-52, 2008.

A. H. Jessen, T. Mikosch, G. Samorodnitsky, *Prediction of outstanding payments in a Poisson cluster model*, Scandinavian Actuarial Journal, Vol. 3, pp. 214-237, 2011.

C. Jordan, *Mémoire sur les formes bilinéaires*, J Math Pure Appl, Vol. 19, pp. 35-54, 1874.

B. Jørgensen, *Exponential Dispersion Models*, Journal of the Royal Statistical Society, Series B, Vol. 49/2, pp. 127-162, 1987.

R. N. Kahn, *Risk and Return in the U.S. Bond Market: A Multifactor Approach*, Advances and Innovations in the Bond and Mortgage Markets, Probus Publishing, 1989.

M. Kearns, *Thoughts on Hypothesis Boosting*, unpublished manuscript, 1988.

D. Kiesenbauer, *Main Determinants of Lapse in the German Life Insurance Industry*, North American Actuarial Journal, 2012.

C. Kim, *Policyholder Surrender Behaviors under Extreme Financial Conditions*, Korean Journal of Applied Statistics, 2010.

K. Larsen, *Generalized naive Bayes classifiers*, SIGKDD Explorations, Vol. 7/1, pp. 76-81, 2005.

S. Levantesi, M. Marino, A. Nigri, F. Perla, S. Scognamiglio, *A Deep Learning Integrated Lee-Carter Model*, Risks, Vol. 7/33, pp. 1-16, 2019.

Life Insurance Agency Management Association (LIAMA), *Factors Affecting Persistency of Orphan Business*, 1948.

Life Insurance Agency Management Association (LIAMA), *The Persistency Raters*, 1949.

N. Liu, *A comparative study of principal component analysis on term structure of interest rates*, JSIAM Letters 2, pp. 57-60, 2010.

J. Liu, J. Qi, L. Wang, Y. Wang, Y. Yu, *An effective and efficient hierarchical K-means clustering algorithm*, International Journal of Distributed Sensor Networks, Vol. 13, 2017.

S. Loisel, V. Maume-Deschamps, X. Milhaud, *Surrender triggers in life insurance: what main features affect the surrender behavior in a classical economic context?*, Bulletin Français d'Actuariat (Institute des Actuaires), Vol. 11/22, 2011.

R. Lord, A. Pelsser, *Level-slope-curvature: fact or artefact?*, Applied Mathematical Finance, pp. 105-130, 2007.

T. Mack, *Distribution-free calculation of the standard error of chain ladder reserve estimates*, ASTIN Bulletin, Vol. 23/2, pp. 213-225, 1993.

M. D. Martinez-Miranda, J. P. Nielsen, R. J. Verrall, M. V. Wüthrich, *The link between classical reserving and granular reserving through double chain ladder and its extensions*, Scandinavian Actuarial Journal, Vol. 5, pp. 383-405, 2015.

McKinsey&Company, *Winning Share and Customer Loyalty in Auto Insurance. Insights from McKinsey's 2012 Auto Insurance Customer Insights Research*, pp. 3, 2012.

C. R. Nelson, A. F. Siegel, *Parsimonious modelling of yield curves*, Journal of Business, Vol. 60, pp. 474-489, 1987.

F. Nogueira, *Causal Inference in Practice with Observational Data*, 2016.

Organization for Economic Cooperation and Development, *Retention Ratio (Non-Life)*, 2016.

J. F. Outreville, *Whole Life Lapse Rates and the Emergency Fund Hypothesis*, Insurance: Mathematics and Economics, 1990.

P. Parodi, *Computational intelligence techniques for general insurance*, Accepted SA0 research dissertation, Institute and Faculty of Actuaries (IFoA), pp. 51-53, 2009.

F. D. Patrick, A. Scobbie, *Some Aspects of Withdrawals in Ordinary Life Business*, Transactions of Faculty of Actuaries 31, 1969.

K. Pearson, *On lines and planes of closest fit to systems of points in space*, Philos Mag A 6, 559-572, 1901.

R. Pelessoni, L. Picech, *Some applications of unsupervised neural networks in ratemaking procedure*, General Insurance Convention & ASTIN Colloquium, pp. 549-567, 1998.

G. Piatetsky-Shapiro, *KDnuggets Methodology Poll*, 2002.

G. Piatetsky-Shapiro, *KDnuggets Methodology Poll*, 2004.

G. Piatetsky-Shapiro, *KDnuggets Methodology Poll*, 2007.

G. Piatetsky-Shapiro, *KDnuggets Methodology Poll*, 2014.

G. Piatetsky-Shapiro, *KDnuggets Software Poll*, 2016.

M. Pigeon, K. Antonio, M. Denuit, *Individual loss reserving with the multivariate skew normal framework*, ASTIN Bulletin, Vol. 43/3, pp. 399-428, 2013.

X. Piulachs, R. Alemany, M. Guillen, *A joint longitudinal and survival model with health care usage for insured elderly*, UB Riskcenter Working Papers Series, Vol. 2, 2014.

Produktinformationsstelle Altersvorsorge, *Basismodell der Produktinformationsstelle Altersvorsorge (PIA)*, .

I. Puri, *Using Machine Learning to Predict Interest Rate Changes from Federal Reserve Proceedings*, written for CS299 course (Machine Learning), Stanford University, 2016.

R. Rebonato, *Interest Rate Option Models*, Wiley, 1998.

C. F. B. Richardson, J. M. Hartwell, *Lapse Rates*, Transactions of Society of Actuaries, Vol. 3/7, 1951.

D. T. Russell, S. G. Fier, J. M. Carson, R. E. Dumm, *An Empirical Analysis of Life Insurance Policy Surrender Activity*, Journal of Insurance Issues, 2013.

R. Sanche, K. Lonergan, *Variable reduction for predictive modeling with clustering*, Casualty Actuarial Society Forum, pp. 89-100, 2006.

R. E. Schapire, *The Strength of Weak Learnability*, Machine Learning, Kluwer Academic Publishers, Vol. 5/2, pp. 197-227, 1990.

W. M. Schmidt, *Interest rate term structure modelling*, European Journal of Operational Research, Vol. 214, pp. 1-14, 2011.

G. E. Schwarz, *Estimating the Dimension of a Model*, Annals of Statistics, Vol. 6/2, pp. 461-464, 1978.

G. Shmueli, N. R. Patel, P. C. Bruce, *Data Mining for Business Intelligence*, Wiley, 2010.

R. Stanton, *Rational Prepayment and the Valuation of Mortgage-Backed Securities*, Review of Financial Studies, Vol. 8, 1995.

L. E. O. Svensson, *Estimating and Interpreting Forward Interest Rates: Sweden 1992-1994*, IMF Working Paper, Vol. 94/114, 1994.

G. Taylor, G. McGuire, J. Sullivan, *Individual claim loss reserving conditioned by case estimates*, Annals of Actuarial Science, Vol. 3/1-2, pp. 215-256, 2008.

R. Trippi, E. Turban, *Neural networks in finance and investing*, McGraw-Hill, 1996.

R. J. Verrall, M. V. Wüthrich, *Understanding reporting delay in general insurance*, Risks, Vol. 4/3, pp. 25, 2016.

H. Wu, *Global stability analysis of a general class of discontinuous neural networks with linear growth activation functions*, Information Sciences, Vol. 179/19, pp. 3432-3441, 2009.

M. V. Wüthrich, *Covariate Selection from Telematics Car Driving Data*, SSRN Manuscript ID 2887357, 2016.

M. V. Wüthrich, *Machine learning in individual claims reserving*, Swiss Finance Institute, Research Paper Series 16-67, 2016.

M. V. Wüthrich, *Neural networks applied to Chain-Ladder reserving*, SSRN Manuscript ID 2966126, 2017.

M. V. Wüthrich, C. Buser, *Data Analytics for Non-Life Insurance Pricing*, SSRN Manuscript ID 2870308, 2018.

M. V. Wüthrich, M. Merz, *Stochastic claims reserving manual: advances in dynamic modeling*, SSRN Manuscript ID 2649057, 2015.

J. Yao, *Clustering in Ratemaking: Applications in Territories Clustering*, Discussion Paper Program, Casualty Actuarial Society, pp. 170-192, 2008.