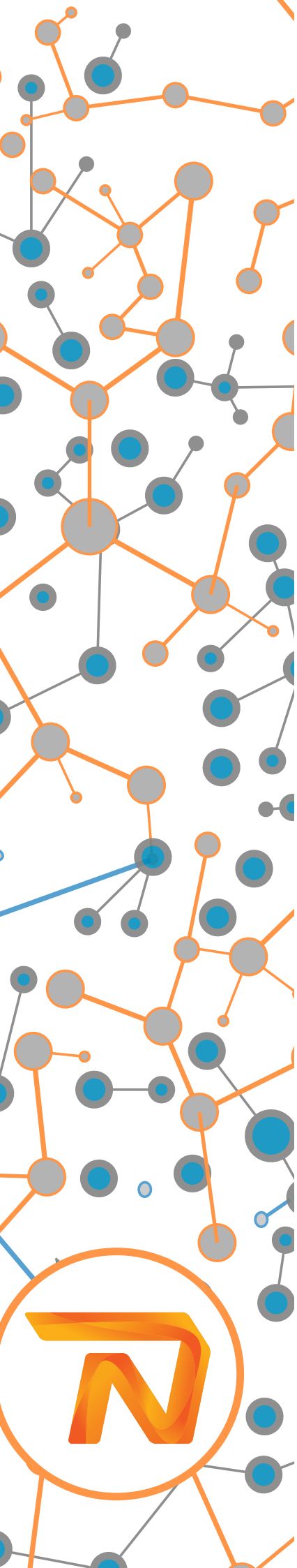

UNIVERSITY OF TWENTE.

**Faculty of Behavioral,
Management & Social Sciences**

DISABILITY INSURANCE

Predicting accurate inflow probabilities in an imbalanced data setting

T.J. Speldekamp

M.Sc. Thesis

October 2018

Supervisors:
Dr. B. Roorda
Dr. R. Joosten

University of Twente
The Netherlands

Preface

Conducting this research has helped me improve my understanding of applying the theoretic skills I learned at the university into a real life project. Without the assistance and involvement of my supervisors, employees of Nationale-Nederlanden and my friends and family, this would have not been possible. I would like to use this opportunity to express my gratitude to everyone who supported me throughout this process.

I would like to thank my primary supervisor Berend Roorda for the excellent guidance and to occasionally remind me to think smaller. Your tip to first explore simpler models instead of immediately implementing complex ones proved to be very helpful. I also want to express my appreciation towards Reinoud Joosten from the University of Twente. Your attention for detail while reviewing my report has really improved the scientific quality of the research.

During my time at Nationale-Nederlanden many people have helped me improve my understanding of the insurance industry in general and the possibilities *and* difficulties of using data within a large organization. I would like to especially express my gratitude towards Hanneke van Veldhuizen and Martine de Gier. Martine de Gier, your knowledge on actuarial pricing and the product was essential to the project. Furthermore, your enthusiasm and positive attitude always sparked good conversations and proved to be very motivating. Hanneke van Veldhuizen, the results obtained would not have been possible without your thorough guidance. Our meetings helped me stay on track and your insights and critical view allowed me to provide a report I am proud to present. At moments I doubted this, you always figured out to a way to express the positives and push me to think about improvements.

Finishing my studies required more than academic support alone. I would like to thank my parents, brother and sister as, although my studies and activities caused me to be away from home quite a bit, I could always count on their support. I would like to thank all my friends for making my student time a period in which, besides studying, I gained valuable new and exiting personal experiences. Finally, I would like to thank my girlfriend for her support and encouragement.

Concluding, I thank you, the reader, for taking the time to read this report.

Management summary

A core business of insurance companies is to set a premium for the policy holders. To set an accurate premium, one needs to model the probability of the insured to produce a claim. In the disability insurance branch for high income employees in the Netherlands, this poses a problem. Similar to other insurance products, the amount of claim versus non-claims is imbalanced, leading to difficulties in setting correct probabilities. At the moment, generalized linear models are widely used to estimate the probability in the insurance branch. Possible reasons are the high interpret-ability and the imbalance in the data of most insurance products. Machine learning models often assume a balanced distribution between classes and the outputted probability estimates do not always give a good estimates of real probabilities. In our research we examine the effects of using machine learning models on a extremely imbalanced dataset. The use case is the ”Arbeids Ongeschiktheids Pensioen” (AOP) product of Nationale-Nederlanden.

Internal data is available on employees and contract basis, for the years 2008 till 2016. The dataset contains 1436 claims for the 765993 records available. The current AOP pricing model incorporates the yearly salary and age of employees in order to obtain the probability. A key component of the research was to explore the possibility to improve predictions by adding external and internal data to the model. We add external data features based on the geographical locations of employees and the companies for which they work.

We evaluate five machine learning models in the research based on 232 features in total. We implement a decision tree, a neural network, the Adaboost algorithm and a Gradient boosting algorithm. To review the performance of a linear model we include the logistic regression model. Both the class separation, between claims and non-claims as well as the accuracy of probabilities needed to be assessed. To this end, we use evaluation metrics. We show that certain, often applied, metrics such as the ROC-curve, can give misleading results in these extreme imbalanced cases. We therefore apply a precision-recall curve and its area under the curve to assess class separation. Furthermore, we note that empirically checking multiple metrics and visualizations is needed in order to evaluate a model based on predictive accuracy.

By assessing the evaluation metrics we conclude that all considered models had difficulties in separating the claim from non-claim data. The precision-recall curve of most models showed that in order to predict 50 percent of the claims correctly, 18 percent of the non-claims where predicted as claim, which implies around 125000 records in the training set. Although class improvements in class separation are minimal, all models improve over

the current model. Of the non-boosting methods the neural network provided a good class separability in relation to the logistic regression and decision tree. However, due to the activation function used, probabilities were pushed towards zero and one. The same behavior, although in a less extreme manner, could be observed in the Adaboost model. This model uses the exponential loss function. This loss function penalizes increasingly wrong prediction exponentially. We conclude that using loss and activation functions which increase the output of a model through large gradients can hurt the accuracy of probability estimates obtained from such a model.

Data re-balancing methods, especially the over-sampling method SMOTE, can improve class separation. However, data re-balancing methods change the distribution of the data the model trains on. When applying set model on test data, the probability estimates became less accurate. We show that by re-balancing the data, probabilities tend towards the new ratio of the claim class. We apply a method to transform the probabilities. However, this method can still result in worse probability estimates. We conclude that re-balancing data when the intent is to directly use the probabilities resulting from machine learning models often results in inaccurate probabilities.

A Gradient boosting model which optimizes the logarithmic loss function provided adequate class separation and accurate probabilities. The area under the precision recall curve increased from 0.00409 in the current model to 0.02076 for the Gradient boosting model. The mean squared error between the predicted probabilities and the true occurrences, also denoted as the Brier score, decreased by 0,00000726 over the 76650 records in the test set. The intent is not to separate the classes using a threshold. However, as an illustration we see a decrease of the number of false predicted non-claims to 15 percent, instead of 18 on more basic models, if we wish to predict 50 percent of the claim cases correctly. These improvements, although seemingly small, can lead to large difference over the many records considered.

In order to obtain the features with predictive power among the large feature subset, we implement a Genetic Algorithm designed for feature selection. Using this algorithm proved to increase performance. As the algorithm can be applied to any model, future machine learning research within Nationale-Nederlanden could benefit from this model. The downside to the Genetic Algorithm model proved to be the duration. Especially for the boosting algorithms, with a long training time. The current model used only two features in order to obtain probabilities. Through visualizations and in our research produced dashboards, designed to extract the inner workings of the Gradient boosting model, we conclude that the added features, such as geographical and contract related data, can improve the probability estimates.

The main recommendations are to apply the produced Gradient boosting model and use the produced dashboards in order to review shortcomings of the old model. The addition of more features and the performance of a more advanced machine learning technique to predict probabilities should provide motivation to start similar projects within Nationale-Nederlanden. Finally, we emphasize the need for strict guidelines in data gathering in order to decrease the time to create a suitable dataset.

Contents

List of acronyms

WIA	”Werk en Inkomen naar Arbeidsvermogen”
IVA	”Inkomensvoorziening Volledig Arbeidsongeschikten”
WGA	”Werkhervatting Gedeeltelijk Arbeidsgeschikten”
WGA	”Werkhervatting Gedeeltelijk Arbeidsgeschikten”
AOP	”Arbeids Ongeschiktheids Pensioen”
UWV	”Uitvoeringsinstituut WerknemersVerzekeringen”
VPU	”Voorziening Periodieke Uitkering”
NN	Nationale-Nederlanden
LE	last earned
CS	current salary
Rvc	”Restverdiencapaciteit”
MW	minimum wage
WULBZ	”Wet Uitbreiding Loondoorbetalingsverplichting Bij Ziekte”
CAO	”collectieve arbeidsovereenkomst”
LR	Logistic Regression
DT	Decision Tree
NN	neural network
SVM	Support Vector Machines
SMOTE	Synthetic Minority Over-sampling Technique
RUS	Random under-sampling
GA	Genetic algorithm
ROC	Receiver Operating Characteristics
PR	Precision-Recall
FN	false negative
FP	false positive
TN	true negative
TP	true positive
FPR	false positive rate
TPR	true positive rate
BS	Brier score
AUC	area under curve
AdaB	Adaboost
GB	Gradient boosting

Chapter 1

Introduction

For working citizens the risk of a career-ending disability poses a significant economic security threat [1]. Most countries have some sort of public program targeted at persons with disabilities, including social insurance benefits [2]. In the Netherlands the ”Werk en Inkomen naar Arbeidsvermogen” (WIA) legislation is in place to cope with working disabilities. During the first two years after an incapacity for work occurs, an employer is obligated to continue the payment of the employee. If after these two years the employee is still incapable to work, the employee enters the ”Werkhervatting Gedeeltelijk Arbeidsgeschikten” (WGA) or ”Inkomensvoorziening Volledig Arbeidsongeschikten” (IVA). In this period, after the first two years, the employee is insured to a maximum of 70% of the social insurance pay (SV-loon) which is equal to 54.617 Euros in 2018 [3].

Like many other Dutch insurance companies, Nationale-Nederlanden issues a product to reduce the relapse in income of employees earning more than the maximum SV-loon in case of a disability. In the case of Nationale-Nederlanden, this product is the ”Arbeids Ongeschiktheids Pensioen” (AOP) product. The AOP product covers the income which exceeds the maximum SV-loon. The insured amount covers 70% of the difference between the last earned salary of the employee and the maximum SV-loon. It is activated when the work incapacity lasts more than 2 years. The product is incorporated in the Dutch Pension Law, meaning disability insurance for employees earning more than the ”SV-loon” is mandatory.

A core business problem of insurers is to set the premium for the policyholders. As the insurance market gets more competitive and efficient it is advantageous to charge a premium corresponding to the expected loss of a particular policyholder [4]. This expected loss is based on the probability of an event occurring and the loss if it occurs. The probability of an employee, incorporated in a product like the AOP product, claiming is very low. Hence, accurately setting a premium for this product poses a complex problem.

This research is conducted for the Damage & Income department within the non-life branch of Nationale-Nederlanden. Here the team Data & Analytics is looking to improve the pricing model of the AOP product in close cooperation with the team Pricing & Actuarial Support. Due to the low probability of a claim, more advanced machine learning techniques, might prove beneficial in predicting the disability probabilities.

1.1 Problem statement

Since 01-01-2014, the pricing for the AOP product consist of three main components. One of these is the inflow probability into the WIA or IVA regulation. This probability is calculated through general and some internal data. Probabilities are based on a differentiation between age and salary of the insured. The second part consists of the predicted claim level and disability percentage and the third component is contract based. The claim level is predicted through the expected ”Voorziening Periodieke Uitkering” (VPU) which covers the benefits of incurred and reported claims. This is based upon the age, average mortality age and a certain rate based on the chances of rehabilitation. Furthermore complicating the pricing is the possibility for employees to only be partially work disabled meaning they can still work. Finally, for the AOP product an employee needs to be incapable of working after 2 years, for which the probability is low.

The current inflow probability is calculated through generalized data and differentiation is based on multiple large clusters. For instance, within salary parameter two classes are separated by a threshold being ”bovenwettelijk” and ”onderwettelijk” (above or below the maximum “SV-loon”). As such, the inflow probabilities are not as differentiated for different insured employees as might be optimal. The current pricing scheme has been the standard since the beginning of 2014 as it was unclear whether improvements could be made. Traditional prediction models and methods have proved to be insufficient to accurately predict the inflow probability. Among the reasons that these models failed is the imbalanced distribution among policyholders that claim and those that do not. Furthermore, the resulting probabilities should be fair for different policy holders.

1.2 Motivation

The interest of Nationale-Nederlanden to introduce machine learning and other advanced data analysis techniques, to improve their income related product pricing, led to investment in researching the possibilities of improving the pricing of the AOP product. Nationale-Nederlanden has comprehensive internal data for the AOP product, more so than for other products. Using more data and predicting the inflow into the IVA and WIA regulation of employees more accurately would improve the pricing model of the AOP product. Within Nationale-Nederlanden, and most insurers, most pricing models are build up out of Generalized Linear Models and more advanced, high dimensional techniques are not yet widely used among actuaries [5]. This research might provide the necessary motivation to start with more extensive and differentiated data gathering and the use of advanced models for other products within Nationale-Nederlanden.

1.3 Objective and research questions

A clear goal should be understandable, useful, testable and incremental [6]. The last two insure that we can see whether we have accomplished what we set out to do, at the end and during the research. With these requirements in mind we set our research goal as,

More accurately predict the yearly inflow probabilities of AOP insured employees into the WIA or IVA regulation.

Achieving this goal leads to an improved pricing model for the AOP product. To accomplish this goal we set the following research question,

How can we utilize machine learning techniques to, more accurately, predict the inflow rate of individual AOP insured employees into the WIA and IVA regulation, thereby improving the pricing model of the AOP product of Nationale-Nederlanden?

We formulate the following sub-questions to divide the research question into distinct parts.

1. How can we select the features in the (internal) data corresponding to the highest predictive power in relation to entering the WIA/WGA regulation?
    - How should one handle very unbalanced data with respect to feature selection and producing a training and test set?
    - How can we determine the importance of different features and the impact of their values on the inflow probability?
2. Which machine learning models are applicable for predicting disability rates and which model performs optimal?
    - How do we define a measure for the model performance?
    - What machine learning models perform well under unbalanced data?
    - Which models produce accurate probabilities?

1.4 Methodology

Within the research we base our methodology on the “Knowledge Discovery in Databases” methodology of [7] often applied in gaining knowledge through machine learning. It contains the following chronological, but often reiterated, steps:

1. **Initial dataset creation and background analysis**
   We research the context of the AOP product in depth and define a clear view of the current pricing model as to develop an understanding of the application domain. Thereafter, we merge and visualize data. This process is described in Chapter 2.
2. **Find literature on pre-processing imbalanced datasets**
   Find literature relating to feature selection, missing value handling and models used for disability insurance and similar problems with unbalanced data. Furthermore, a performance measure of the model needs to be established. The literature used in our research is outlined in Chapter 3.
3. **Missing value handling**
   Deletion of possible errors and handling of missing values in the data. This process is summarized in Chapter 4

4. **Data transformation**
   Feature selection (removing features) and data transformation for feature extraction (adding features). Furthermore, we normalize or scale values accordingly additional to creating a training, validation and test set. These steps are reported in Chapter 4.
5. **Find Literature on machine learning models**
   Find literature relating to machine learning tasks and algorithms applicable to the situation. We again refer to Chapter 3 for the outlined literature.
6. **Construct the (machine learning) model(s)**
   Choose the appropriate machine learning task (regression, classification etc.). After which we select one, or multiple, machine learning models and train it, or them, on the training data. This process is described in Chapter 5.
7. **Evaluate the model on the found performance measure.**
   Evaluate the model on the performance measure and compare performance of the new and old AOP model. As the evaluation and training of models is an iterative step we also outline this step in Chapter 5.
8. **Refine knowledge through results**
   This step includes the identification of important features and their relation to the predictions. We refer to Chapter 6 for these results.
9. **Draw conclusions and recommendations**
   Answer the main research question defined in Section 1.3, provide recommendations and discus possible drawbacks of the research. We report our findings in Chapter 7.

1.5 Ethical considerations

The introduction of the General Data Protection Regulation (GDPR), during the course of this research, helps to illustrate the importance of data protection in large entities, such as Nationale-Nederlanden. The research conducted incorporates sensitive data associated with private individuals and thus data protection has been of importance. To this end, in deliberation with a Data Protection Officer, the data has been anonymized. This is done through,

1. Values in the data have been converted to representations which can not be traced back to its original form without a certain “decoding” file.
2. This “decoding” file is password protected and saved in a secure network environment.
3. Some column names of the dataset containing sensitive data have been changed.

Furthermore, the use of personal data in order to improve the premium of individuals requires some deliberation. Throughout the research ethical considerations with respect to using certain variables, have been discussed openly, and responded to accordingly.

Chapter 2

Context description

In this chapter we outline the context in which the research is conducted. The Dutch legislation relating to working disability and the AOP product, including its current pricing scheme, is outlined in respective Sections 2.1 and 2.2. The available internal data is outlined in Section 2.3.

2.1 The Dutch disability legislation

In order to improve the pricing of the AOP product, the structure of the legislation concerning work disability in the Netherlands and the different kind of benefits an employee can receive, should be clear.

An employee who is at least 35% disabled for work and remains so, two years after his working disability occurs, is eligible for either IVA or WGA. These are the two benefit legislations in the WIA [8]. Whether an employee receives a benefit, and which kind, is based on the disability percentage and the chance of recovery. The maximum benefit an employee can receive is coupled to the ”SV-loon”. This is the salary of an employee over which taxes and social premiums are paid [9]. Each year a maximum is set to this ”SV-loon”. In 2018 this was set at 54616,80 Euro a year [3].

Before an employee enters the WIA legislation he receives his benefit through his employer via the ”Wet Uitbreiding Loondoorbetalingsverplichting Bij Ziekte” (WULBZ). The benefit received during these two years is 70% of the last earned (LE) salary.[1] At the end of two years after a disability occurred, the UWV assesses the disability percentage. The employee enters the IVA or WIA legislation if this disability percentage is higher than 35%. If not, an employee does not receive any benefits and is expected to continue working.

2.1.1 IVA legislation

If an employee is 80% or more disabled and chances of recovery are non-existing, he enters the IVA legislation. The benefit received, is equal to 75% of his LE salary [10]. However, this is maximized at 75% of the maximum SV-loon.

[1] In the first year this can be more if this is incorporated in the ”collectieve arbeidsovereenkomst” (CAO)

2.1.2 WGA legislation

If an employee is more than 35% and less than 80% disabled to work, or has a chance of recovery, he enters the WGA legislation. The benefit structure is salary dependent in a period of time following the first two years. The duration of the period depends on the employment period at the company where the work incapacity occurred. As the employee is only partially disabled he can still have a current salary (CS). The WGA legislation makes a distinction between employees which earn more or less than 50% of their ”Restverdiencapaciteit” (Rvc). The Rvc is calculated through multiplying the disability percentage of an employee with the LE salary.

In the salary dependent period the WGA benefit does not differ if the Rvc is earned and is based on 70% of the difference between the LE salary and an employees CS. This is, again, maximized at 70% of the maximum ”SV-loon”. In the continued benefit period the WGA legislation does make a distinction between earning more or less than the Rvc amount. If an employee earns less, the continued benefit is based on 70% of the minimum wage (MW) amount plus his CS (which is less than his Rvc). If an employee earns more, the continued benefit is based on 70% of the difference between his LE salary and his Rvc, plus his CS (which is more than his Rvc. Again, this is maximized at 70% of the maximum ”SV-loon”.

2.2 The “Arbeids Ongeschiktheids Pensioen” product

As mentioned, the benefits for employees are maximized at 70% of the maximum amount of ”SV-loon”. Employees earning more will experience a relapse in income after a disability to work occurs. To reduce this relapse, the AOP product is offered. The benefit from the AOP product is based on 70% of the difference between an employees LE salary (higher than the maximum ”SV-loon” amount) and the maximum ”SV-loon” amount. This benefit is multiplied by the disability percentage. The ‘extra’ benefit is added to the benefits already received through the WIA and other insurance products.

The product is incorporated in the Dutch Pension Law, meaning disability insurance for employees is mandatory. However, employers are not obliged to use the AOP product from an private insurer such as Nationale-Nederlanden. If they decide not to, they can expect the risk of financing the benefit themselves. Furthermore, employees need to deal with the ”poortwachterswet”. This requires that employees continuously monitor the health of their employees and report this to the ”Uitvoeringsinstituut WerknemersVerzekeringen” (UWV). Paying premium for the AOP product redistributes this record-keeping to the insurer. The insurance is financed by a pay as-you-go premium.

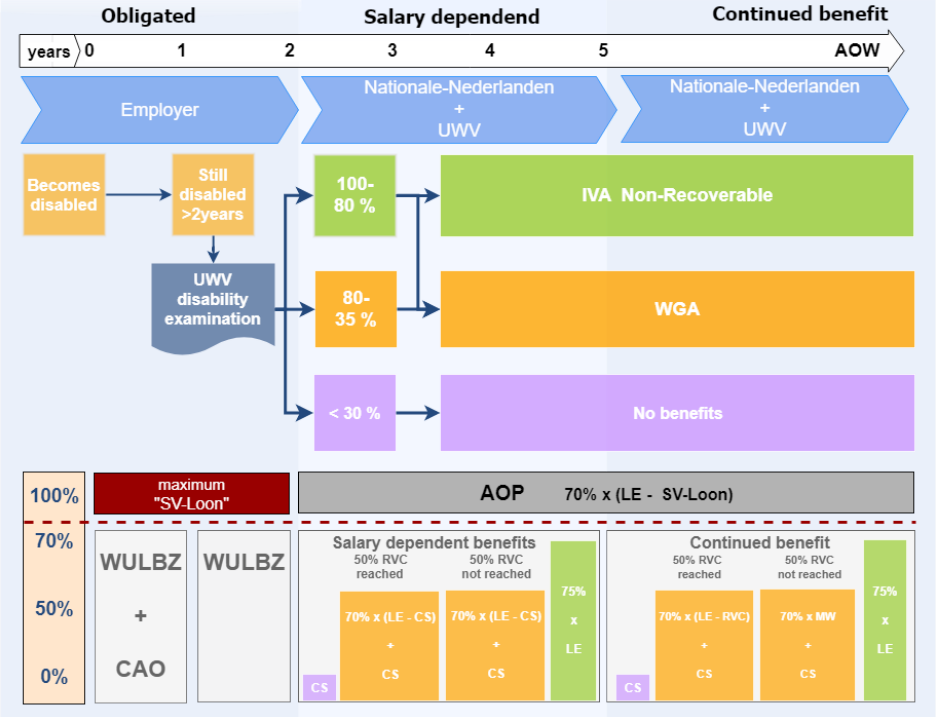

Figure 2.1: Timeline and benefit in the Dutch disability system.

2.2.1 Current pricing model

In the current pricing model three main components for employees are taken into account. One part being the expected claim level (CL) and predicted disability percentage (DP). The other part is the probability of inflow into the WIA legislation of the individual employees. The premiums are calculated on a contract basis, meaning for multiple employees an employer pays a collective premium. This is the third component of the pricing model. The gross premium (GP) can be calculated through [11],

$$GP = \frac{P_D * DP * CL}{1 - cost\% - commission - CoC}.$$

The estimated inflow probability is denoted by P_D. The terms in the numerator refer to extra revenue needed for commissions, costs and required internal yields on a product.

Claim level

The claim level, or "schadelast" is the expected incurred claim amount that Nationale- Nederlanden needs to cover for this particular account. It is calculated on a employee level basis and is incorporated in the VPU. The VPU is based on the yearly salary (often above the maximum "SV-loon"), as well as possible maximum insured amounts an employer decided on when signing the contract.

Inflow probability

The focus in this research is on the inflow probability. The inflow probability is currently based on data obtained from "Verbond van Verzekeraars" and generalized linear models applied to a small subset of internal data [11]. "Verbond van Verzekeraars" is the association for non-life and life insurance in the Netherlands [12]. Based on extensive data they report probability numbers of events happening for different common insurance products.

The current inflow probability estimate for entering the WIA or WGA legislation is based on salary and age. The salary component is differentiated, only, upon being greater or less than the maximum "SV-loon" of the year the premium is set, causing a discontinues inflow probability at each age level. The current inflow probability estimates for some different age and salary levels is displayed in Figure 2.2.[2]

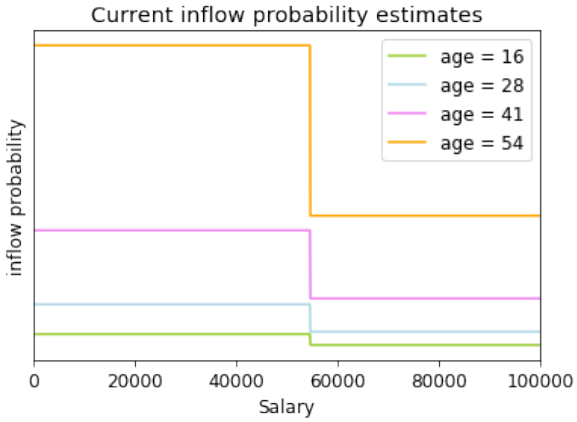

Figure 2.2: Inflow probabilities of the current model.

Contract level

On a contract level basis, a premium for a contract can be adjusted based on the number of employees for which the product is in place, a certain risk premium and provision for Nationale-Nederlanden.

2.3 Available data

This section describes the available internal data. We describe some basic analyses and show the size and distribution of the datasets. Internal data is available on a contract, employee and claim level basis. Contract and employee data is stored in the PICO database whereas claim data is stored in the ASPRO database.

2.3.1 Contract level data

On a contract level, data is available on a contract containing at least three employees of one firm. Contract data is stored in the PICO database. The type of data fields considered are described in Table 2.1. The scope of the research is to predict claims of the AOP product

[2] We do not display the actual inflow probability due to the sensitivity of this company specific information.

and as such only contracts of this product are considered and basic data not applicable to the scope, e.g. contact telephone info, are removed.

Table 2.1: Contract level information.

Feature type	Examples	# Features
Contractual agreements	Starting & end dates, payment clause, minimal insured amount etc, etc.	14
Pricing agreements	premium tariff base, discount on contract etc.	5
Company data	geographical data, number of employees, (possible) umbrella organisation, etc.	11
Database information	Contract number, broker number, etc.	4

In total we have 34 contract related features. In Appendix B.2 the individual features are listed. The data format is mixed and ranges from binomial to numeric. In total we have the data for 14777 different contracts which were at one point active during the period 2008 till 2018. In the last few years many new contract where signed, which is reflected in the number of employees displayed in Figure 2.3.

2.3.2 Employee level data

The employee data is essential in this research as we try to more accurately predict the occurrence of a claim based on differentiated employee information. The employee data includes data on participating employees within each contract. The primary key connecting the employees to a specific contract is the contract number. Employee data is, like the contract data, stored in the PICO database. A short description of the data can be found in Table 2.2. The raw data can be found in Appendix B.2.

Table 2.2: Employee level information.

Feature type	Examples	# Features
Employee data	Geographical data, gender, age, salary, AOW age, etc.	12
Insurance information	Insurance type, a risk class, etc.	5
Database (NN) information	Contract and employee number, status (disbarred) etc.	6

In total there are 98764 unique employees. To obtain a dataset applicable for training a machine learning model, employee data from different years had to be merged to claim and contract data. As we want to have, for instance, the correct yearly salaries and ages of employees at that years contract data for employees who work multiple years. The merging process is described in more detail in Appendix A. In Figure 2.3, we visualized the number of employees active during the years 2008-2017. As we can see the number of employees within the AOP product has been increasing over the years.

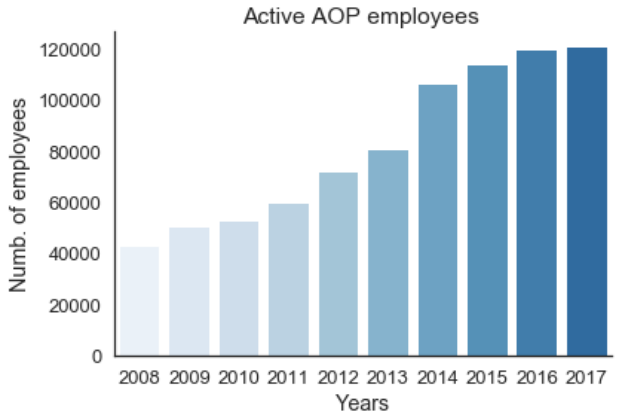

Figure 2.3: Active AOP employees over the year 2008 - 2017.

2.3.3 Claim level data

The claim data is stored in a different data warehouse than the employee and contract data. Claim data can, in theory, be merged on the claim numbers back to the individual contracts and the employee for which the claim was emitted. In practice there are serious complications. Within the employee data the claim numbers have been added manually and as such the claim number may not always be identical or even missing in the employee data. Furthermore, the employee number, which is in the claim data as well as the employee data, is manually added in the claim data. Hence this key is also not always identical or available. Although integral to the research we lay out the merging process and its difficulties in Appendix A.

Claim data, stored in the ASPRO data warehouse, contains the height of the payed out claims and the starting date of the payments among others. As most records do not contain claim data, these features are mainly used for correctness of the merging process. We discard the features for prediction purposes. The occurrence of a claim is kept, as this is our target for prediction.

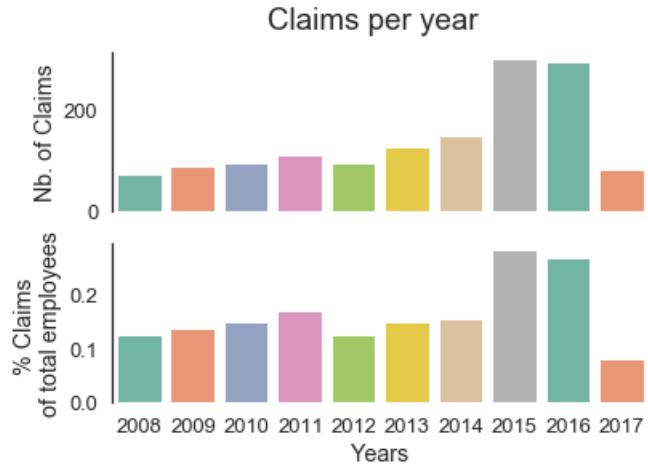

Figure 2.4: Number of claims over the years 2008 - 2017.

As we can see in Figure 2.4 the data on the claims is extremely imbalanced. From the 98764 records of employee data there is only data on 338 claims. This might cause problems as the data for which we try to predict the in-and outflow probabilities is in the minority in comparison to the data with no claims. Furthermore, we see an increase in the number of claims from 2015 onward. This can be attributed to early reporting of a working disability by employers. However, due to the AOP product only paying out when an employee is two years disabled to work, multiple cases might not eventually claim and return to work before the product becomes active. The data from 2017 is often not yet updated in ASPRO and PICO. As such, the number of disabilities (not yet AOP claims) is small in this year.

In Figure 2.5 the distribution among salary and age differentiated by claims made or not and gender is displayed. The claims are primarily centered around the max SV-loon boundary and often claims are made for people earning less. Furthermore, older employees seem to more often claim than young employees. Subsequently, although more males are present in the data Figure 2.5 seems to indicate a difference among male and female employees in terms of claiming, especially in terms of age. Finally, we note that besides imbalance in the data between claims or no, there is also no clear distinction among the claims and the non-claim in terms of these feature values. The records with claims are distributed among the non-claim records in terms of the features gender, age and salary.

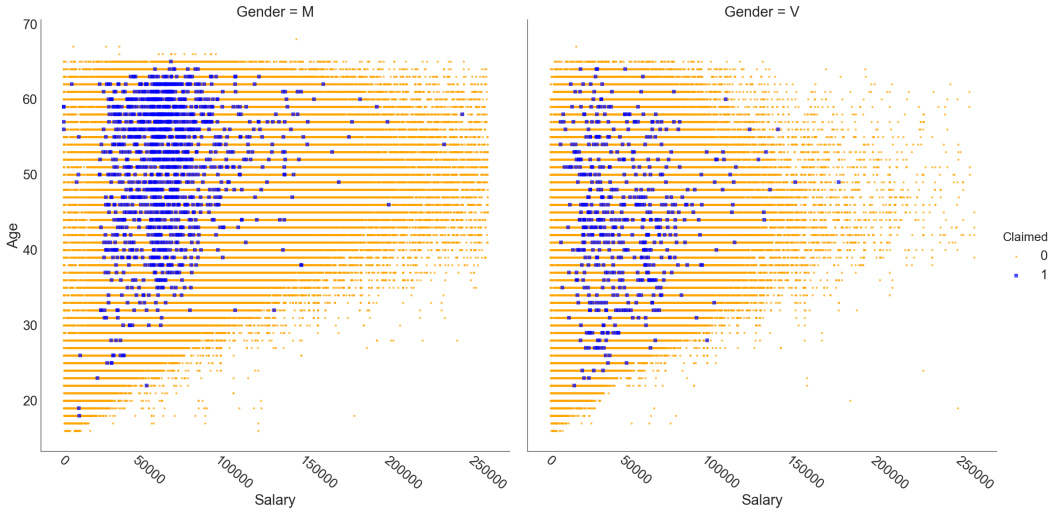

Figure 2.5: Distribution of salary and age.

Chapter 3

Literature

To predict future disability rates it is customary to use GLM models sometimes combined with time series forecasting [13], [14]. The parameters of the GLM are based on maximum likelihood, and are estimated on historical data. The main reason for using GLM based models is the easy interpret-ability with reasonable computational speed [14]. More advanced machine learning models have been applied, with success, in the insurance sector [15], [16], [17]. However, the previously conducted research is mostly focused on car insurance and fraud detection. Possible reasons for this focus is due to the large amount of claims in car insurance. Furthermore, the conclusions often refer to accuracy in predictions and only involve a slight mention of the predicted probabilities.
This research focuses on estimating probabilities for an insurance product for which the claim occurrence is very small. This leads to an imbalanced dataset. We focus on using machine learning models in order to predict the probability of a claim occurring. To this end, we review literature on a machine learning process in general in Section 3.1. Where after, in Section 3.2 literature on dealing with an imbalanced data distribution is covered.

3.1 General machine learning process

In this section tasks corresponding to a machine learning process in general are covered. Necessary steps such as the handling of the dataset, deciding on the machine learning task and possible algorithms are reviewed.

3.1.1 Null-value handling

After obtaining a dataset, the first step of the data pre-processing stage is dealing with missing values [7]. Most algorithms assume a complete dataset is present and therefore require data preparation and cleaning for useful knowledge extraction [18] [19]. The selected method for dealing with missing values can have a dramatic impact on data interpretation and hence has to selected carefully [20].
The most basic and common way of dealing with missing values is list-wise deletion (delete rows/columns with missing values) [21] Although a common method, it might cause im-

portant information to be discarded which may be detrimental, especially in imbalanced datasets. Using a method to fill in missing values is known as missing value imputing [18]. Basic ways of imputing are mean or median imputation. For a certain variable the mean or median of that variable over all records is imputed for the missing values.
Another method of value imputing uses the k-Nearest-Neighbor (kNN) algorithm. It is a popular technique as it is intuitive. However, it requires a distance function between records that can deal with categorical and numerical features [18]. Based on a distance function between variables, the k closest neighbors of records with missing data are found. The missing data is imputed for a certain feature based on the values of this feature of the neighbors. An advantage is that it requires no advanced parameter tuning, however the value for k (number of neighbors) needs to be determined empirically [18].
A method introduced in recent years uses the Random Forest algorithm for imputing missing values. A Random forest is able to deal with mixed-type data and is non-parametric [22]. [22] compares this imputation technique, labeled MissForest, to kNN and the Expectation-Maximization algorithm and find that MissForest can handle complex high-dimensional datasets with ease and still provide excellent imputation results.
In [23], four methods of imputing were compared based on how they approximated the original distribution of a dataset from which values were removed. From imputing methods zero (imputing 0), median, minimum, and kNN they draw the conclusion that "kNN imputation was the best method for restoring data statistics" [23]. In a similar test, [24] concludes that a MissForest and an algorithm similar to Expectation-Maximization [3] had significantly lower errors than kNN.

3.1.2 Machine learning with a probabilistic prediction task

In [7], defining a machine learning task revers to the type of algorithms to use and primarily the produced output. Common machine learning tasks include classification and regression. However, in some applications, it is more important to establish the probability of an event occurring. One of such application is the probability of a claim occurring based on certain characteristics. Although regression might seem appropriate for defining a certain rate between zero and one, the true probabilities of these events occurring are often not known. Hence, no appropriate training data for regression is available. Machine learning algorithms used for classification can often output, besides the predicted class, a conditional probability of a certain record belonging to a certain class. An application in which these conditional probabilities are used for establishing a rate, not unlike disability prediction, is default probability (PD) prediction. For PD prediction, a number of studies have shown the advantages of using machine learning systems over more traditional approaches [25].

[3] Expectation-Maximization is an iterative procedure based on a Maximum Likelihood Estimator. Although effective, we leave it out of discussion as extensive parameter tweaking needs to be executed to fit the distribution of the data.

3.1.3 Classifier methods

The methods mentioned to handle the class imbalance problem all require the eventual prediction, or conditional probability, outcome of a base classifier, or the combination of a multiple base classifiers. We denote algorithms as a base classifier if it composes a single classification algorithm without changes to the algorithm itself in terms of cost-sensitive modifications.

Linear models Linear models, such a Logistic Regression (LR), can be written in the form,

$$g(x) = w_0 + \sum_{i=1}^{N} w_i x_i.$$

Hence, we approximate the outcome of the model g by finding optimal parameter values $(w_0, w_1, .., w_N) \in \mathbb{R}$ [26]. A linear model solves the classification problem with a hyperplane. An immediate disadvantage is that only linear separability is possible. However, if it can obtain satisfactory results, linear models are widely accepted in practice and the weights w_i are interpretable. Furthermore, the prediction of linear models are stable, resulting in a low variance.

Decision tree A Decision Tree (DT) is a simple, but widely used, model that partitions the input space by vertical and horizontal boundaries into M regions. The outputted DT can be viewed as a sequential process in which at each 'node' a decision (i.e. $Feature_x > threshold$) is made. The feature on which the decision, or 'splitting' occurs needs to be assessed as well as threshold on which to split. As this might be computationally demanding, often a greedy optimization is done by starting with a single begin (root) node and then growing the tree by adding nodes one at a time [27]. These added nodes are assessed based on a certain splitting criterion such as the Gini index or entropy. The advantages of a DT are its interpretability, speed and ability to handle continuous and discrete data [28]. DTs are however, linear in its decision boundaries and can be prone to over-fitting. Furthermore, DTs often exhibit high variance in their predictions as a small change in data can result in very different splitting points from which the change is propagated through the tree [26].

Neural Network A neural network (NN) can be described as a series of transformations through a series of basis functions Φ. Different units (neurons) are connected through a certain weights w, similar to parameters w of the linear models. The basis function Φ_1 for a single layer network has the form,

$$\Phi_1(x) = h(w_{1,0} + \sum_{i=1}^{N} w_{1,i} x_i).$$

Here h is the activation function which is often non-linear [26]. Neurons are connected to an in- and output layer of the network. The output is obtained by forward-propagating the input values based on the weights in the network (a series of transformations).

Learning occurs by back-propagation, in which the weights are updated to correct for errors in the output layer [27]. Advantages of NNs include the possibility for linear and non-linear classification and that they can include continuous and discrete data, although the input needs to be scaled properly. Disadvantages include, the lack of interpretability of the trained weights, the many different network structures and the slow training speed if a large amount of data is used.

Possible other methods include Support Vector Machines (SVM), Naive Bayesian classifiers and k-NN [26]. The k-NN algorithm searches for neighbors and bases classification upon those. However, in a very imbalanced setting the number of neighbors whose true class is of interest is low. A SVM is kernel based and thus is more appropriate for classification instead of probability estimation. The attempts to place the SVM in a probabilistic framework are still relatively undeveloped [29]. Naive Bayesian classifiers assume independence between features. Although sometimes the result of this classifier, considering this assumption, is still good, we do not consider the algorithm.

3.2 Dealing with the class imbalance problem

Classical machine learning methods prefer that the number of records in the different classes is similar [30]. However, as the science of machine learning matured and was applied in real world problems, the class imbalance problem became more apparent as the imbalance caused suboptimal classifier performance [31]. Class imbalance refers to data in which we have a class with a considerably higher number of records corresponding to one class, which we call the majority class, as the rare or the minority class [32]. It is often the case that the minority class is the one we are most interested in. The reasons for the suboptimal classifier performance are, among other, classifiers trained for good coverage on the majority set, minority samples being treated as noise and the other way around, overlapping distribution regions of the classes and by definition, the minority class has little samples which might be problematic in training the model [33] [34]. One can mitigate class imbalance issues in multiple stages of the machine learning approach. In the data-preprocessing stage through re-sampling the data, reducing the noise in the data by feature selection and extraction, one-class learning and by applying cost-sensitive learning in the preprocessing stage [33] [35]. In the algorithmic stage one can apply ensembles of classifiers, bagging, boosting and modify the basic algorithms parameters [33]. Furthermore, ill defined performance measures can have a serious impact on the training and validation process.

3.2.1 Data pre-processing solutions

Data re-sampling

Data re-sampling refers to removing or adding samples to re-balance the dataset. Within re-sampling we can perform under-sampling by eliminating majority-class samples and over-sample by duplicating minority-class samples [36]. Sampling can be random or by more intelligent methods. Re-balancing the data is a popular strategy [33]. However, [36] notes that

random oversampling might lead to overfitting on duplicated minority samples and under-sampling could lead to discarding important majority samples. More intelligent sampling methods include Synthetic Minority Over-sampling Technique (SMOTE) [33], cluster based sampling, or using advanced machine learning algorithms such as an SVM to re-balance the data [33] [36] [35]. SMOTE is a technique where new synthetic minority samples are created by drawing a distribution between two existing minority samples and randomly picking a place within this distribution to place a new point [37]. This method and adoptions have been proven successful in handling imbalanced datasets in practice. However, [36] does warn that even intelligent oversampling might violate the equivalence between the distribution before and after re-balancing, stating that re-sampling has "drawbacks that other methods, such as cost-sensitive learning, do not have, if implemented properly" [36].

Feature selection

One of the reasons for suboptimal performance due to class imbalance is, as mentioned, the presence of noise or minority records interpreted as noise. Feature selection, or selecting a subset of features from the entire feature space, can reduce the noise and increase classification performance [33]. Feature selection techniques can be organized into filter methods, wrapper methods and embedded methods depending on how they combine the feature selection search with the construction of the classification model [38].
Filter methods refer to techniques which select features purely based on a single features values. A feature is removed based on a certain score such a the correlation between the feature and the decision variable. Advantages of filter methods include computational efficiency. However, filter methods ignore interaction with the classifier and features are often considered separately and as such do not take into account the correlation between different features corresponding to classifying the decision variables [38].
Wrapper methods do take into account the interaction between the classifier and the feature selection process. Wrapper methods have a certain search procedure in which multiple possible feature subsets are defined and evaluated based on a evaluation metric trough the the classification result of a trained classifier. Wrapper methods do include dependencies between the features but have a risk of overfitting and are computationally intensive as we need to train a different classifier model on several feature subsets [38]. Wrapper methods can use an heuristic (deterministic) or a randomized search method. Heuristic search methods include techniques such as forward an backward search which are relatively simple but might get stuck in local optima. Randomized search methods include the Genetic algorithm (GA) (evolutionary) and randomized selection. These methods are less prone to local optima but are more computationally intensive [38] [33].
A third class of feature selection methods are embedded methods. Here the feature selection process is embedded within the classifier. An example here might be a decision tree in which features with less predictive power are not considered anymore. Embedded methods are far less computationally intensive than wrapper methods but are even more specific to the classification model and as some algorithms can not incorporate this [38].

3.2.2 Algorithmic solutions

In the algorithmic stage, solution for the class imbalance problems are aimed toward "strengthening the learning process towards the minority class" [39]. This can be done by modifying the algorithm itself, or ensembles of basic classifiers can be used.

Modifying the algorithm often comes down to adapting the training process to become cost sensitive. On an algorithmic level, cost-sensitive learning applies to modifying the training objective function to penalize errors in the minority class more [33].

Ensembles

Ensembles of classifiers refer to using multiple base classifiers to be trained on the training data and using these separate classifiers to predict on the test data. Creating an ensemble consists of two parts. One is to select the distinct base classifiers and the second part consist of defining a way to combine the predictions of those classifiers into one prediction per test-set record [40]. In regards to selecting base classifiers, one should select individual classifiers that performing well [4] on the training data. Additionally, individual classifiers should be diverse, otherwise an ensemble structure will not lead to improved results. As one can imagine, having multiple homogeneous classifiers which all perform well, but all produce the same errors on certain records will not produce a better result if combined. Creating diversity in the classifiers does not necessarily mean, using different types of base classifiers. Although using for instance a Neural Network, a decision tree and a k-NN is a possibility, creating diversity in data sampling or classifier parameters can also provide the diversification needed [40].

There are several ensemble methods often used in practice. These are Bootstrap Aggregation, Boosting and Stacked Generalization [41]. Bootstrap Aggregation, often referred to as Bagging, is a simple yet effective method in which a number T classifiers are trained on different bootstrapped (sampling with replacement) subsets of the data [41]. Then, often through majority voting, predictions are combined. Random Forests are also a form of a bagging ensembles. Here, vertical bootstrapping is also done by bootstrapping random subsets in the feature subspace.

Boosting also uses a form of data re-sampling. However, instead of bootstrapping, the weights of data records are adjusted based on whether or not a record was predicted correctly [40]. The next classifier will be trained on the same data sample for which weights for the samples have been adjusted. The weights for 'harder' records will be increased through each iteration when they are misclassified often. In the first, well received boosting algorithm, AdaBoost [42], predictions are combined via weighted voting based on the training error achieved in each iteration. Boosting algorithms can significantly increase the performance, but can degrade with noise in the data [40].

Stacked Generalization is a ensemble method which primarily differs in how individual classifier predictions are combined. After an ensemble of classifiers has predicted values for

[4] Performance should be at least above randomly classifying records. However, the definition of adequate performance is based on the data that is used.

a certain test set. A 'meta classifier' is trained on this predicted output. This meta classifier then serves as a way to optimize the combination of predictions and can correct improper training of a particular classifier in the ensemble [41]. Although this method performs well, the complexity, both in terms of time and interpretability, is high [40]. Furthermore, a second decision has to made regarding a suitable meta classifier.

Probability calibration

Ensemble algorithms can often obtain a higher accuracy then simpler algorithms. However, due to the structure of some ensemble algorithms the outputted probabilities yield "poor squared error and cross-entropy error" [43]. This is due to badly calibrated posterior probabilities. Using probability calibration methods one can adjust outputted probabilities. Methods include Platt scaling [44] and Isotonic Regression [43]. Both methods include training a model on a separate part of the training data, which is used to transform the predicted probability outputs of a model.

3.2.3 Performance evaluation

As one deals with imbalanced datasets, standard performance evaluation methods do not apply. One might consider the accuracy of a model. If only 0.2 percent of the data is of the class one would like to predict, gaining a, misleadingly high, accuracy of 99,8 percent would be obtained by predicting all records as the majority class. Hence, when evaluating models on a imbalanced dataset we require a metric which can look at the accuracy on both classes. Most performance metrics for classification are based on a confusion matrix, as presented in table 3.1.

	Predicted C_1	**Predicted** C_0
Actual C_1	TP (true positives)	FN (false negatives)
Actual C_0	FP (false positives)	TN (true negatives)

Table 3.1: Confusion matrix

For imbalanced data, frequently adopted performance metrics are based on a sort of dispositions towards false negative (FN)s and false positive (FP)s [39]. One of these methods is the Receiver Operating Characteristics (ROC) curve. A curve is composed by calculating the false positive rate (FPR) and true positive rate (TPR) and plotting it, for different decision thresholds on the conditional probability of the test set records for the different classes [39]. The FPR and TPR are respectively, $\frac{FP}{FP+TN}$ and $\frac{TP}{TP+FN}$. Although handy for visualizing prediction evaluation over multiple models, the ROC curve does not give a certain score on which to evaluate. To this end the Area Under Curve of the ROC (AUCROC) is used. The AUCROC is 1.0 in a perfect model and 0.5 when equal to picking randomly between

classes. Although widely popular, the ROCAUC score might be misleadingly high in extreme class imbalance and might be inconsistent (produce the same results) among classifiers with different performance [45].

An second, less applied, method is the Precision-Recall (PR) curve (PRC). The idea is similar to the ROC. The precision and recall are plotted for multiple thresholds. Here precision is obtained through $\frac{TP}{FP+TP}$ and recall by $\frac{TP}{FN+TP}$. The main difference between the ROC and PRC are that the ROC includes true negative (TN)s while the PR-curve does not. The area under curve of the PR-curve (AUCPRC) is 1 in a perfect model and equal to the minority class percentage in a random model.

As the aim is to predict probabilities, evaluation metrics such as the $F1 - Score$ are not considered. The F1-score incorporates the rates obtained from the confusion matrix (FPR and TPR) directly and hence, only incorporates the pure classification results.

The evaluation methods mentioned until now only apply to separability among classes as they incorporate which records where correctly classified. However, as mentioned in Subsection 3.1.2, we are not necessarily interested in the amount of true positive (TP)s and FPs as we intent to predict a certain probability. Hence, although classification results might be a good reference to the predictive value of the model, we need (an) evaluation method(s) to measure the model performance on the probability prediction. The "true" probability of record obtaining a claim or not, is not known. Nevertheless, we can measure the error of the predicted rate in reference to the rate of a claim occurring in our data. We assume the probability to be one if a claim occurred and zero if not. We measure probabilistic accuracy using the Brier score (BS). The BS "is the average quadratic loss on each instance in the test set" [45], or;

$$S_{Brier} = \frac{1}{N} * \sum_{n=1}^{N} (g(x_n) - y_n)^2.$$

Here, $g(x_n)$ is the outcome of the model g on test record n. Hence, the BS quantifies average deviation between predicted probabilities and the outcome of a model [45]. In assessing the accuracy of predicted probabilities, the BS is consistent between classifiers with different performance (a lower BS, for a model, always means probabilities are overall closer to the true occurrence than one with a higher BS).

Another useful method to assess the accuracy of probability prediction is a calibration plot. In a calibration plot the predictions are discretized into a number of bins. Records with predictions which lie within these bins are grouped. Thereafter, the fraction of positive class cases is plotted against the mean predicted value of the records within each bin [43]. Well calibrated probability predictions will lie near the diagonal.

Chapter 4

Data preperation

This chapter contains the data preparation stage of our methodology. In Section 4.1 we deal with missing values in the data. Subsequently, in Section 4.2, we extract new features from existing internal data. Thereafter, we describe external data in Section 4.3. In Section 4.4 we lay out a framework for the training, test and validation split. Concluding, we outline the feature selection procedure we apply in Section 4.5.

4.1 Missing value handling

Nationale-Nederlanden does not explicitly gather data with the intention of using machine learning techniques for prediction. This has led to missing values in the dataset. When collected, some features might have been left blank intentionally. When such a dataset might be used for training a model, these missing values suddenly might take on great significance [46]. As mentioned in Chapter 3, the missing data imputation method can have a large impact on data interpretation [20].

For handling missing values an expert on the AOP product has been consulted about the features in the dataset. On the one hand, this consultation was used in order to select relevant features for the remainder of the process (see Section 2.3). Furthermore, it was used in order to evaluate whether a value was missing at random (MAR) or intentionally left blank. Although it was not always possible to correctly identify whether it was MAR, we opted to fill in -1 or NK (not known) for some missing values in certain columns if they seemed to be intentionally left blanc.

For the remaining columns an imputation technique is used. In Chapter 3 several imputation techniques and some advantages and disadvantages are outlined. Due to the very unbalanced dataset, the current distribution of the data is crucial in predicting the minority samples correctly. Furthermore, due to the small number of minority samples, retaining data is important. On account of these two reasons and comparative studies on imputation techniques outlined in Chapter 3, list-wise deletion and mean/median imputation are not considered. The MissForest algorithm [22] performs well in comparison to multiple other imputation techniques considered in studies. Furthermore, the MissForest algorithm can handle mixed-type data easily (unlike the k-NN imputation method) and can be used for regression and classification purposes. Hence, we opt to use the MissForest algorithm in order to impute missing values.

We follow the algorithm proposed by [22] in order to impute the missing values. We assume a dataset $X_{n,m}, n \in N, m \in M$ with N representing records and M features. For arbitrary records $X_{n,s} \quad s \in M$, that includes missing values in feature s we separate the dataset into four sets. These are,

$\mathbf{y_{n,s(obs)}}$ The observed values of feature $X_{n,s}$,

$\mathbf{y_{n,s(miss)}}$ The missing values of feature $X_{n,s}$,

$\mathbf{X_{n,m(obs)}}$ Features $X_{n,\not s}$ with observations $m(s)$ $obs = (1, , n) \in y_{n,s(obs)}$

$\mathbf{X_{n,m(miss)}}$ Features $X_{n,\not s}$ with missing $m(s)$ $miss = (1, , n) \in y_{s(miss)}$

Table 4.1 is an example table with missing values in three features f_2, f_4 and f_5. If we are looking to impute the missing values in feature f_5, $\mathbf{y_{n,s(obs)}}$ would include y_1 and y_2, $\mathbf{X_{n,m(obs)}}$ includes $X_{i,j}$, for $i \in 1, 2, j \in 1, ., 4$. Then we predict missing values y_3 and y_4 using test set $\mathbf{X_{n,m(miss)}}$ which would include $X_{i,j}$, for $i \in 3, 4$ and $j \in 1, ., 4$ in table 4.1.

Table 4.1: Null value example table.

record\Features	f_1	f_2	f_3	f_4	f_5
1	$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	**NaN**	y_1
2	$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$	y_2
3	$X_{3,1}$	**NaN**	$X_{3,3}$	$X_{3,4}$	**NaN**
4	$X_{4,1}$	$X_{4,2}$	$X_{4,3}$	$X_{4,4}$	**NaN**

Initially we find features with missing values $X_s, s \in m$ an sort by ascending number of missing values in the feature values. Within the set $X_{n,m(obs)}$ & $\mathbf{X_{n,m(miss)}}$ which we use for missing value prediction training and prediction, other feature values might be missing (In f_2 and f_4 in Table 4.1). We define two sets of features. Set C containing the categorical features of the data and Set R containing the numeric features. We impute missing values in $X_{n,m(obs)}$ & $X_{n,m(miss)}$ by the mean, if numeric ($m \in R$), or mode of the feature values if categorical ($m \in C$. Hereafter, we train a random forest classifier [47] on $X_{n,m(obs)}$ to predict $y_{n,s(obs)}$. Using the trained classifier we predict $y_{n,s(miss)}$ using $X_{n,m(miss)}$. We update our dataset X using the predicted values and iterate further over the other features with missing values $X_{n,s} \quad s \in m$. We keep iterating until all $X_{n,s}$ are imputed, using the already imputed values to predict new ones in different features. If all features with missing values X_s are imputed we use the newly imputed values and keep iterating until the stopping criterion v is reached. We define stopping criterion v as [22],

$$v = \sum_{s \in R} \frac{(y^{imp}_{s(miss)} - y^{old}_{s(miss)})^2}{(y^{imp}_{s(miss)})^2} + \sum_{s \in C} \frac{\mathbb{1}_{y^{imp}_{s(miss)} \neq y^{old}_{s(miss)}}}{(\#NA)}.$$

For numerical values we look at the mean squared error between the newly imputed values and the value in the iteration before. We then divide it by the value of imputed values. For categorical values we look at the number of changes made in one iteration (identity $\mathbb{1}$) and divide by original number of missing values. Hence, v describes the degree of change made in each iteration. If stopping criterion v is higher for the next iteration then we stop the algorithm.

As we do not have the "real" missing values in the original data, there is no way to see what the error of the MissForest algorithm is on those records. To test the algorithm we can delete the missing values from a column, here the yearly salary, and from the records left we make a number, here 1500, of artificial missing values for which we know the true value.

Test index	Target	Predicted
0	67974.00	67965.80
1	76022.28	76022.28
2	49297.00	49297.00
3	89700.00	90109.36
...	...	...
1496	53136.00	53218.94
1497	61776.00	61549.15
1498	57236.00	38991.22
1499	59865.72	60378.91

Table 4.2: Predicted yearly salary.

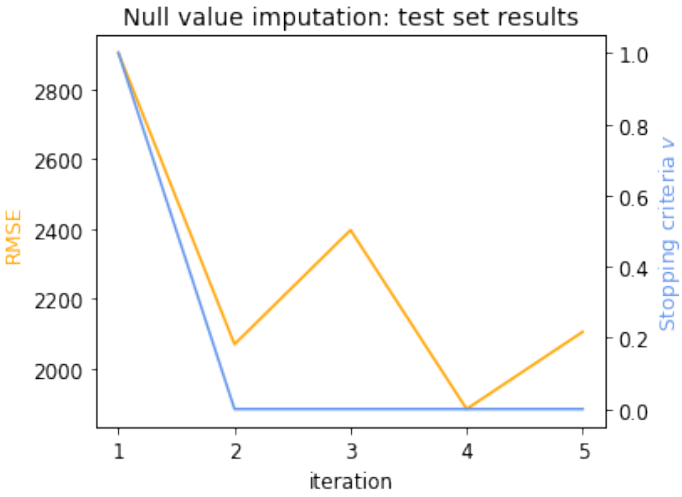

Figure 4.1: RMSE and stopping criterion v.

The results on this synthetic test set are depicted in Figure 4.1 and Table 4.2. The stopping criteria v decreases to almost zero after one iteration, meaning not much is changed in the imputed values after this iteration. However, it does keep decreasing until iteration five. We plotted the Root Mean Squared Error (RMSE) of the imputed (predicted) yearly salaries and the true values. The error is decreasing for most iterations. In iteration three and five the RMSE goes up. This behavior is to be expected as we are not iterating to decrease the RMSE. In Table 4.2 we can see that the predictions are very accurate, although for some, like record 1498, the predicted value can be off. However, this test set seems to imply that imputation using the MissForest Algorithm allows for imputation while keeping the original distribution intact.

4.2 Feature extraction

After handling the null-values we use information already incorporated in the data by means of extraction. This is done by aggregation, creating lag in the data or transformation of feature values. In this research we aggregate over the contract IDs over age and salary and create the features noted in Table 4.3,

Table 4.3: Extracted features by aggregation

Aggregate over	feature name	description	data format/ possible values
CONTRNR SALARY	salary_min	minimal salary within contract	float
	salary_max	maximum salary within contract	float
	salary_mean	mean of salary within contract	float
	salary_median	median of salary within contract	float
CONTRNR AGE	age_min	minimal age within contract	integer
	age_max	maximum age within contract	integer
	age_mean	mean of age within contract	float
	age_median	median of age within contract	integer

Furthermore, we create for instance a category based salary to incorporate "bovenwettelijk" and "onderwettelijk" information. We extract the features presented in Table 4.4. [5]

Table 4.4: Extracted features based on single features

Based on feature	Feature name	Description	Data format/ possible values
INGYR/ MND/DG	DIFF_ING_NU	Time difference between employees begin date in contract and the year of the data	float (years)
EXPIRYR/ MND/DG	DIFF_EXPIR_NU	Time difference between contract expiry date and the year of the data	float (years)
JS-SALARIS	SALAR_CAT	If employee earns more or less then the max SV-loon for that specific year.	AboveSV/ BelowSV
	DIFF_JSAL_WITH _CONTMEAN	Difference between employee salary and the contract mean salary as mentioned in table	float
GBRJR/ MND/DG	LEEFTIJD_DLNR	Age of the employee	int
	DIFF_EINDLF_AGE	Difference between 'AOW' age and age of employee	int
	DIFF_AGE_WITH _CONTMEAN	Difference between employee age and contract mean age as mentioned in table....	float

4.3 External data

The internal data described in Appendix B includes a small number of geographic features. In the current pricing model geographical locations are not incorporated. We add a large number of features based on the postal codes of the companies and the individual employees. The external features are from an external data source, which NN has access to. They mostly represent social percentages based on geographical location. These external fea-

[5] We also created a "lag" in the data to incorperate the salary increase/decrease. However, later in the process it turned out that due to the structure of the merging process this gave the algorithms a method to more accurately predict the claims

tures are described in short in Table 4.5 [6]. Note that we give some examples, however some features like the province might are also included on an employee level.

Table 4.5: External data information

Feature type	Examples	# Features
Contract company level	Province, degree of urbanization, percentage of financial workers in area, etc.	14
Employee level	Percentage of higher educated people, percentage of families, etc.	5
Extracted data	distance from work to living area.	3

4.4 Training, test & validation set

Previous steps have made use of the entire dataset created from the merging process described in Appendix A. From this point forward, we divide our dataset into a combined training/validation set and a separate test set. A machine learning model is trained on training data. Then often, a separate validation set is used to optimize model parameters. This however, results in losing valuable training data (some of our small amount of claims). Hence, we use K-fold cross validation throughout this research on a combined training and validation set. This involves using this combined set and partitioning it into K folds, or groups. Then we train on the $K-1$ other folds and validate on the remaining fold. This process is repeated until each fold has been used as a validation set. The resulting evaluations can be combined by taking the mean. Furthermore, a standard deviation over the folds, in terms of the evaluation metric, can be obtained as well. This allows for a description of the variance of the trained model. Within this research we use 5-fold cross validation.[7] A separate test set is used at the end of the machine learning process to provide an unbiased evaluation of the final model. This is needed, as tuning algorithm parameters based on the validation set potentially leads to over-fitting on this combined training/validation set.

We apply a stratified train/test split. Meaning that in both sets (training set is the combined training/validation set) the proportion of claims occurred is approximately equal. Furthermore, we choose to produce these stratified training/test splits separate for each year in the data. Hence, for each year, a training/validation set is made and a separate test set. These are then combined for all of the years into the final training/validation and test set. This decision resulted from gaining a test set which incorporate claim and non-claim data throughout the different years in the dataset. Furthermore, the year 2017 has been removed as the data for this year was incomplete as mentioned in Sub-section 2.3.3.

[6] We do not explain all external features as there are around 120 which were added in order to see the effect

[7] 5- or 10-fold cross-validation is recommended [48].

4.5 Feature selection

The next data preparation step contains the feature selection or, removing features with no predictive power. In Chapter 3 we outline multiple feature selection methods and refer to advantages and disadvantages applicable to datasets with imbalanced data. Feature selection can be accomplished through three unique methods. Filter methods are computationally efficient but will often not capture the dependencies between data, the feature selection process and eventually the machine learning algorithm. Embedded methods do capture these dependencies but are very specific to a machine learning algorithm. Wrapper methods, although computationally heavy, can capture dependencies among features and can incorporate different, even multiple, classification algorithms. Furthermore, randomized search wrapper methods allow for the global minimum to be found. As a wrapper method for feature selection we implement the GA proposed by [49]. The structure of the algorithm allows for feature selection to be implemented with relative interpretability and GAs have achieved good performance with the feature selection process within other research with imbalanced data [50], [51] .

GAs are known for their ability to efficiently search large datasets with little parameter tuning [52]. A GA is based on the principles of natural selection and genetic mutations. Starting with an initial population composed of individuals, specific selection rules are set to reduce the population to a state that minimizes a certain fitness function [53]. Within the context of feature selection we outline the pseudo GA algorithm in Algorithm 1.

Algorithm 1 Genetic Algorithm - Feature Selection

INPUT: $X_{N,M}, \quad y_N, \quad p, \quad l, \quad v, \quad K, \quad G$ ▷ K, refers to number of K-folds

▷ G, refers to the model used

procedure GENETIC ALGORITHM($X_{n,m}, p, l, v, K, G$)

  Initialize population Z_p, with chromosomes $c_j \quad j \in p$

  while $p > 1$ **do** ▷ stop when only one chromosome left in Z_p

    for $j = 1$, $j{+}{+}$, while $j < p$ **do**

      select $X^j_{n,m'}$, with $m' \in M$ for $c_{j,m'} = 1$

      determine $\mu\ F_j$ by evaluating $G(X^{j,K}_{N,m'})$ over K folds ▷ F refers to the fitness

    normalize F_j over j and sort, set as prob. dist D

    for $i = 1$, $i{+}{+}$, while $i < l * p$ **do**

      Select parents based on D ▷ Based on roulette wheel approach

      Create new chromosome through crossover ▷ Crossover at random int.

      Perform mutation on new chromosome ▷ If rand float $< v$

      Add new chromosome to Z_{new}

  return Z_p ▷ Return population of 1 chromosome

The primary input consist of training dataset $X_{N,M}$ with N records and M features. Feature selection revers to creating a dataset $X_{N,m'}$ in which $m' < M$ [7]. We train on set $X_{N,M}$ based on $y_N \in \{0, 1\}$ in which 1 represents a claim and 0 no claim. We randomly generate a population Z of p chromosomes c_p representing feature vectors. Each chromosome represents a subset of the features $m' \in M$. Each population exist of a list of chromosomes in the form of Table 4.7. We iterate over the chromosomes $c_j \quad j \in p$ in the population and select the features present in each chromosome, represented by a one, from the training set. We then obtain, for each chromosome c_j, a training set $X^j_{N,m'}$. On this set $X^j_{n,m'}$ we train a classifier G and evaluate the performance by K-fold cross validation. The fitness F_j for chromosome j represents the average performance over K folds, based on the evaluation metric used.

After evaluating the performance on each chromosome we apply crossover and mutation to generate a new population of offspring [54]. Crossover is based on a "natural selection" principle. We generate new individual chromosomes from combining 'old' ones. Based on the performance of each chromosome c_j we rank them from highest to lowest.[8] We use a probability roulette approach to select the chromosomes that are used as parents. F_j is normalized over the whole population after sorting. This normalized fitness is then set as the probability D_j for chromosome j. These probabilities are mapped to cumulative segments of a line in which each chromosomes segment is equally sized to its fitness as can be seen in Table 4.6. A random number between zero and one is generated and the chromosome whose segment spans the random number is selected as one of the parents [55]. In Table 4.6 the worst performing chromosome c_8, has a much smaller chance to be selected. Chromosomes with a higher fitness have more chance of being selected as a parent.

Table 4.6: Probability roulette approach for a Decision tree on the data.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
$F_j(AUCROC)$	0.4828	0.4824	0.4673	0.4668	0.4380	0.4262	0.4217	0.2730
Pr_j	0.1396	0.1395	0.1351	0.1350	0.1267	0.1233	0.1219	0.0789
$Pr_j(F \leq f)$	0.1396	0.2791	0.4142	0.5492	0.6759	0.7991	0.9211	1.0000

Table 4.7: Crossover process of chromosomes

chromosome\features	f_1	f_2	f_3	...	f_{m-2}	f_{m-1}	f_m
c_1	1	0	0	...	1	0	1
c_2	0	0	1	...	1	1	0

chromosome\features	f_1	f_2	f_3	...	f_{m-2}	f_{m-1}	f_m
c_{new}	1	0	0	...	1	1	0

[8] If evaluation metric is a loss that needs to be minimized, transform it to a maximization for the roulette wheel approach.

In Table 4.7 we assume c_1 and c_2 to be selected by the roulette wheel approach. A new chromosome c_{new} is then generated based on these two parents. By randomly picking a crossover point, (here somewhere between f_3 and f_{m-2}) parts of the chromosomes are combined to form a new chromosome, c_{new}, which will be in the next population Z_{new} of size $l * p$. Here l is defined as a depletion rate. The last part in the GA algorithm consists of mutation. Mutation applies to randomly "flipping" a feature gene (a chromosome consists of genes) being flipped to zero or one. This again leads to stochastic searching which might find global optima more efficiently. To control the rate of the number of genes in each chromosome who are mutated we define mutation rate v. After one iteration we obtain a new chromosome population Z_{new}. Here we apply the same logic until we obtain a population of one.[9] The chromosome, which defines the features to be used, is returned.

As the GA algorithm is a wrapper method, we apply this algorithm for each different machine learning models G. Hence, the GA algorithm might return different features based on the machine learning model used. This insures a connection between the feature selection process and the machine learning prediction.

[9]Note that intermediate results are returned. Hence, when chromosomes of previous iterations result in a higher fitness (the method is stochastic) it is possible to pick those.

Chapter 5

Model selection

This chapter contains the model selection process. We review several classifiers based on performance evaluation methods described in Section 5.1. We first discuss more basic, "single", classifiers in Section 5.2, where after we apply class imbalance methods to review their effects on the classifiers performance in Section 5.3. In Section 5.4, we outline algorithmic solutions to improve the performance. Concluding, we review the performance of a subset of classifiers on the test set in Section 5.5, where after we propose our final model.

5.1 Performance evaluation

In order to evaluate which model performs best, we require an evaluation metric. In Subsection 3.2.3, metrics for imbalanced learning and probability accuracy have been outlined. The goal of the research is to predict a suitable probability, or rate, for an individual employee based on its feature vector x. This corresponds to predicting the posterior probability $P(claimed|x_{employee})$.

The ROC-curve and PR-curve represent rates obtained from the confusion matrix for different thresholds on the probabilities $P(C_{claimed}|x)$. In Sub-section 3.2.3 we mentioned certain articles in the literature that prefer the PR-curve over the ROC-curve in imbalanced data settings. However, the ROC-curve remains the most popular metric among articles (also including literature relating to imbalanced data).

In Figure 5.1 the ROC- and PR-curves are plotted for artificial datasets. We create three different datasets, each containing 100.000 records, with each a different class distribution. The first dataset has a balanced class ratio, the second a ratio of 80% negatives and 20% positives and the third has a ratio, not unlike the true data in this research, of 99,8% versus 0,02%. In the upper row of Figure 5.1 the normalized distribution of the artificial predictions are plotted. We create a normalized predictions of the data-points having a true positive target (class 1) and a true negative target. Notice that we create the probability predictions in such a way that no threshold can separate the classes perfectly.

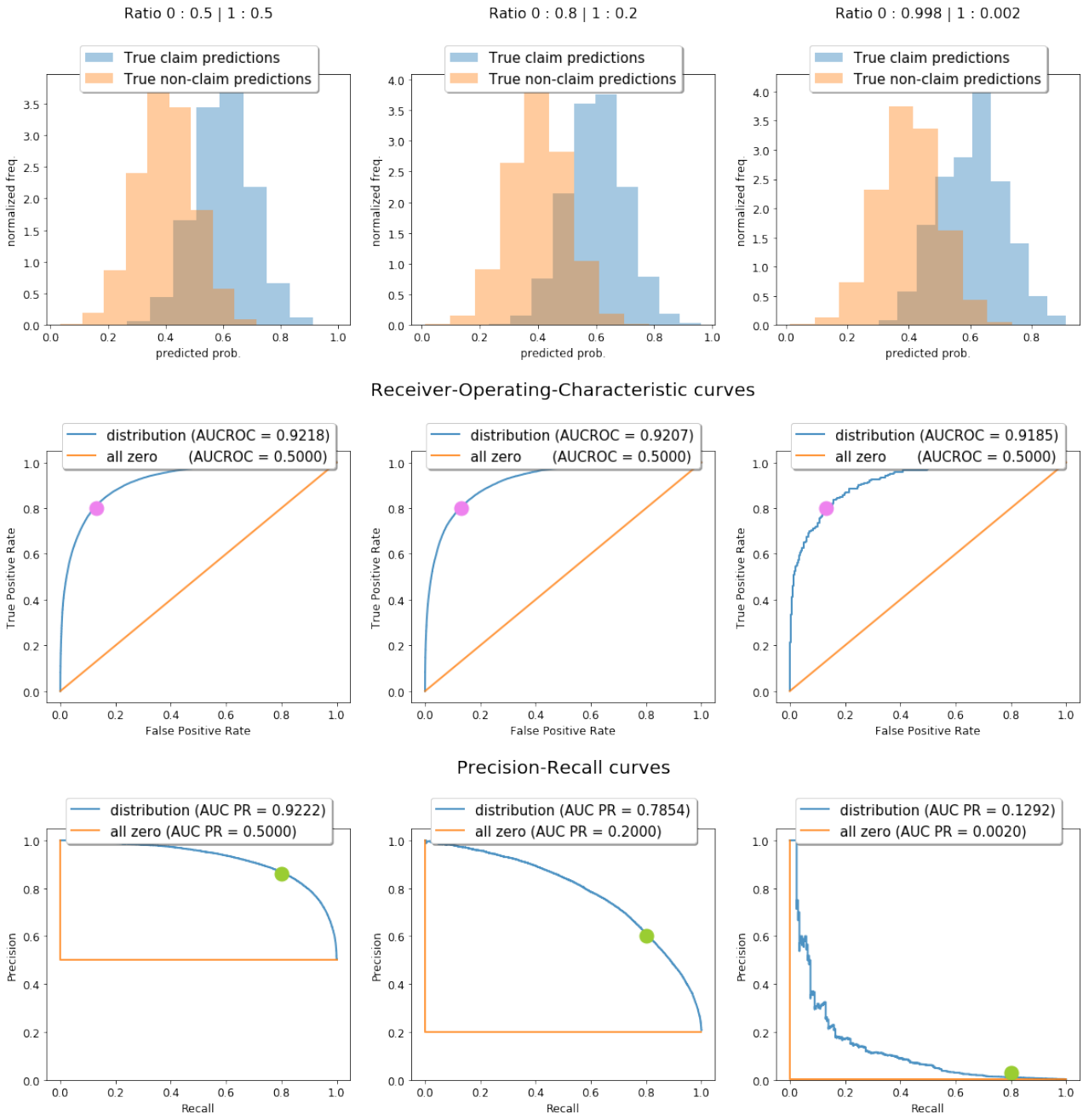

Figure 5.1: Comparing ROC and PR curves.

In Figure 5.1 points are plotted on the ROC-curve at 80% TPR. The TPR ($\frac{TP}{TP+FN}$) in this case has 40.000 TPs (80% of the amount of positives) with 7000 FPs (14% of amount of true negatives) for the balanced dataset. The ROC-curve and in extension the AUCROC, are approximately equal for the imbalanced datasets. However, the number of TPs and FPs for the second dataset are 16.000 and 11.200 respectively. For the third dataset we have 160 TPs and 13.972 FPs. Hence, the ROC-curve, which incorporates the TPR and FPR can not effectively capture the ratio between TPs and FPs effectively in increasingly imbalanced data distributions. Furthermore, as the ROC uses the TNs, the ROCAUC might seem very high mainly due to the large number of negative samples predicted correctly.

When we apply the same analysis (same TPs and FPs) on the PR-curve we have a precision of around 0,85 and a recall of approximately 0,8 for the balanced dataset [10]. For the second dataset we see a decrease in the performance. The precision decreases to 0,6 and in the third dataset the precision approaches zero. Hence, the PR-curve can capture the behavior of imbalanced distributions with respect to the FPs being predicted. The main difference between the metrics is the incorporation of the TNs. In a very imbalanced dataset, the number of TNs can be very high. This leads to the FPs not having a large impact on the FPR of the ROC-curve. Thus the ROC-curve and AUCROC might lead to classifiers being perceived much better even though the number of FPs is much higher. During the research, this behavior has indeed led to misleadingly high AUCROC values. Finally, we note that the perceived performance on a PR-curve can become increasingly worse if the imbalance of the data is increased.

In our research, we require the number of TPs to be high, while the FNs should be relatively low (we do not want to many claims being predicted as non-claims). Hence, we consider recall to be most important. We would allow the FPs to be relatively higher. An amount of non-claims being predicted as claims would not be a serious problem and could even be leading to a conclusion that some records have a large probability of a future claim. However, as we could see in the ROC-curve analysis, due to extreme imbalance the FPs might very much exceed the TPs without the ROC-curve and AUCROC showing this behavior. Furthermore, we are not interested in the TNs, which the ROC-curve does incorporate. Due to the reasons above we use the PR-curve and its area under curve to represent the performance of classifiers. From this points onwards, we refer to the area under the PR-curve as area under curve (AUC).

The PR-curve and the AUC do not express the accuracy of the pure predicted probabilities. The performance of those metrics is based on setting different thresholds for these probabilities and thus only estimates the correct ranking of the predicted probabilities. To assess the accuracy of the probabilities we apply the Brier Score. Finally, we also sum up the predicted output of the model and compare this to our true claim "probability". By marginalizing out x we should have,

$$P(y) = \sum_{n=1}^{N} P(y|x_n).$$

Hence, we compare the total sum of predictions with our summed target variable. This should be similar to one another. Finally, we also review the calibration plot described in Sub-section 3.2.3.

[10] Note that, although other names are used the TPR of the ROC-curve equals the recall of the PR-curve. We use these different terminology of the metrics in accordance with literature.

5.2 Base classifiers

Within a machine learning process it is often the best approach to start learning 'simpler' and interpretable algorithms. When these base classifiers already obtain results that are satisfactory, there is no need to construct complex algorithms. In 3.1.3 we outline some base classifiers, often applied in machine learning processes, which are applicable to our data.

5.2.1 Decision tree classifier

A DT has three type of nodes. A root node, which has no incoming edges, internal nodes with one incoming edge and two or more outgoing edges and leaf nodes, which have only one incoming edge. Root and internal nodes consist of decisions involving features. Leaf nodes include the predictions of the model. The base heuristic implies that when not all records are predicted as one class after a decision threshold, new internal nodes are created. When all records are predicted as one class then this is a leaf node. However, as this will lead to over-fitting often tree pruning and for instance splitting into no decisive class label is done [28].

Decision tree algorithms need to provide a method to asses on which attribute to split, what the decision threshold is and how records are classified after the decision threshold. To measure the best split and in extension on which attribute to split on, a impurity measure is used [28]. Three known impurity measure are the Gini-index, entropy and classification error. The impurity measure is calculated before and after the split. The difference, or the gain in impurity, is then used as a determination for how well this certain split is. Studies have shown that, due to consistency among them, the choice of impurity measure has little effect on the performance of a decision tree [28]. Hence, in correspondence to methods later introduced, we use the entropy as impurity measure which is optimal at 0.5 and useless at zero. In each decision node d, the entropy impurity is defined as,

$$Entropy_d = -\sum_{k=0}^{K-1} \hat{p}_{d,k} * log_2(\hat{p}_{d,k}).$$

With $\hat{p}_{d,k}$ as,

$$\hat{p}_{d,k} = \frac{1}{N_d} \sum_{n=1}^{N} I(\hat{y}_n = k). \tag{5.1}$$

Here k represent the class with $k \in 0, 1$ (binary classification). Hence, for each class k we look at the proportion of records correctly classified. The parameters besides the impurity measure are a tree depth of seven and a minimum amount of records in the leafs of 51. More information on these parameter setting is explained in Appendix D.1

5.2.2 Neural Network

As mentioned in Chapter 3, a NN is constructed out of multiple nodes. The input values are forward-propagated through a network based on weights. The output, produced in the last layer of the network, results in a certain error. The error leads to updating of the weights in the network through back-propagation. In Figure 5.2 we depict a NN structure.

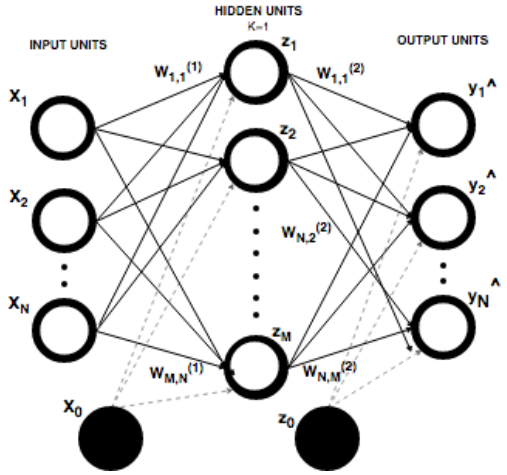

Figure 5.2: Example of a neural network.

The network depicted in Figure 5.2 has one hidden layer, an input and an output layer. The input layer consist of our feature vectors x_n and the output layer consist of our predictions $\hat{y}_n$. Furthermore, we have hidden units $z_m, m \in M$, with M being a user specified setting for the amount of neurons in the hidden layer k. Furthermore, we have weights w between the different neurons in each layer. We represent this, for some of the weights, in Figure 5.2 with $w_{i,j}^k$. Here k represents the layer and the weight is connected from neuron i to neuron j. To get prediction output $\hat{y}$ we use the forward propagation through the network by,

$$\hat{y}(x, w) = h_2\left(\sum_{j=1}^{M} w_{n,j}^{(2)} h_1\left(\sum_{i=1}^{N} w_{i,j}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right).$$

Here h_k is the activation function used to transform the output of each neuron in the layer. These activation functions can include the logistic sigmoid, hyperbolic tangent or a simple linear activation function (see [27]). Although seemingly complicated the output is generated by a series of transformations controlled by a vector of parameters w [27]. Neurons x_0 and z_0 are bias neurons. They allow for a fixed offset in the data to be taken into account [11].

Once obtaining the output y_n for each output node we need to define a certain differentiable loss function L. We use the binomial logarithmic loss error function of the form,

[11] As an example, in the case of a linear combination (a simple line) we apply the bias to have the possibility of the line not passing through the origin.

$$L(\hat{y}; y) = -\sum_{n=1}^{N} \Big(y_n * log(\hat{y}_n) + (1 - y_n) * log(1 - \hat{y}_n)) \Big). \tag{5.2}$$

We want to update the weights to reduce the loss L. This is done through back-propagation. As this loss function is continuous on w we can apply differentiation and we desire $\frac{\partial L}{\partial w}$ to be zero. Due to a lot of weights and the complexity of the loss function $L(w)$, we use numerical optimization using gradient descent [27]. Hence, we iterate multiple times where each time the weights of iteration τ, w_τ, are calculated. Then for the next forward-propagation we use weights $w_{\tau+1}$ with,

$$w_{\tau+1,n,j} = w_{\tau,n,j} - l * \frac{\partial L}{\partial w_{\tau,n,j}}.$$

The gradient of the loss $\frac{\partial E}{\partial w_\tau}$ depends on the loss function used, as well as the activation functions h used within the neural network. In the research we use a network structure of (20,20,20) and use the hyperbolic tangent function in all layers expect the output layer. The reasoning behind these parameters is described in Appendix D.2

5.2.3 Logistic regression

In an LR model, we obtain the conditional probability for the positive class $P(y = 1|x)$ as a logistic sigmoid acting on a linear function of the feature vectors $x_n \quad n \in N$ as,

$$P(y = 1|x_n) = \sigma(w^T x_n),$$

which we will call σ_n [27]. Within the context of our research we have a binary "classification" task. Hence, $P(y = 0|x_n) = 1 - P(y = 1|x_n)$. The $\sigma()$ is the logistic sigmoid function [12]. The maximum likelihood function to optimize w given x and the target $y, \quad y \in \{0, 1\}$, is,

$$L(y|w) = \prod_{n=1}^{N} \sigma_n^{y_n} (1 - \sigma_n^{y_n})$$

To get an error function we take the negative logarithm of this function. This is equal to,

$$E(y|w) = -\sum_{n=1}^{N} \sigma_n * log(y_n) + (1 - \sigma_n) * log(y_n) \tag{5.3}$$

By taking the derivative with respect to w (keeping in mind that the derivative of the logistic sigmoid function equals, $\sigma(1 - \sigma)$) (see Appendix C.1.1), we have for the loss function L,[13]

$$\frac{\partial E}{\partial w} = \sum_{n=1}^{N} (\sigma_n - y_n) * x_n. \tag{5.4}$$

This needs to be zero in order to get the minimal loss. We iterate by gradient descent to get the optimal parameter vector w. Note that LR directly optimizes the logarithmic loss function. This leads to well calibrated probability predictions. We do not use regularization or any other parameters for the LR model as we want to examine the predictive power of a linear model on the data.

[12] $P = \frac{1}{1-e^{-z}}$

[13] Note that in this case $\hat{y} = w^T x$, for which the derivative over w has to be taken as well.

5.2.4 Output of the base classifiers

In Figure 5.3 the PR-curves and corresponding AUC of the LR,NN and DT model are plotted. We plot only the lower left part of the PR-curve due to the seemingly bad performance of all models.[14] All models have trouble separating the small number of claims from non-claim data. The recall is very low if we desire to obtain a relatively high precision. In the setting of our research, this results in a high number of employees with no claim to be predicted as a claim (precision) when one wants to achieve a large number of correctly predicted claims.

The DT model is the model performing worst in term of discrimination among classes. The precision drops for very low values of the recall, meaning too many points are predicted as a claim. The PR-curve shows that the NN can reach a relatively high precision over the range of different recall values. However, in order to obtain a recall of 0.3, the precision is around 0.005 (see the NN and LR result). This would lead to 380 TPs, 913 FNs and 125000 FPs (of the 688050 FPs). Note that the precision is very sensitive as a 20% increase in the precision decreases the FPs by 16% which results in far less FPs with the number of minority samples. Hence, to have a large number of true positives to be predicted as positive (recall), the predicted positives are often a wrong prediction (precision). All models do perform better than a random model. However, we see that if we desire a very high recall and predict 80% of the claims correctly, we are almost on the precision level of a random model. The random line at a recall of 0.8 means that we would also predict around 80% of the non-claims as claims. Thus that prediction would be of little use. The NN achieves the best class separation followed closely by the LR model. The better result of the NN, implies that the data can be better predicted using non-linear functions.

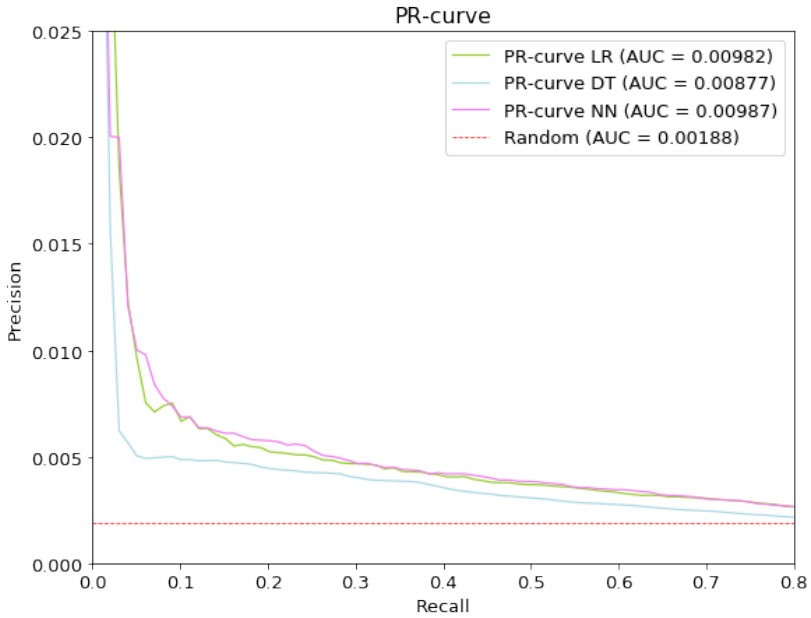

Figure 5.3: PR-curve with AUC of the DT, NN and LR.

[14] A perfect model lies in the upper right corner of a PR curve.

As mentioned in Subsection 5.1, we also review the accuracy of the predicted probabilities. In Table 5.1 the mean and standard deviation of the BS, as well as the sum of the predicted conditional probabilities $\sum Pr(y = 1|x)$ of the models are summarized. Notice, that the BS for the DT model is high compared to the others. The sum over conditional probabilities however is in line with what we would expect. Although the BS of the NN is low, $\sum Pr(y|x)$ is deviates most with the original sum of claims in the data.

Table 5.1: Correctness of predicted conditional probabilities.

$model$	μ_{Brier}	σ_{Brier}	$\sum Pr(y = 1\|x)$	$\sum y$
LR	0.0018689	0.0000033	1294.6	1293
DT	0.0018771	0.0000041	1292.6	1293
NN	0.0018703	0.0000026	1257.3	1293

To get an insight into the meaning of these numbers we review the calibration plot and distribution of predicted probabilities displayed in Figure 5.4. In Figure 5.4 we bin the probabilities and compute the true probability as the percentage of records in the bin having a claim as target variable. The marker size presents the relative amount of records in the bin. As the vast majority of the resulting probabilities are in the range [0,0.1] we plot the calibration in this range. Furthermore, we display the calibration in the range [0,0.003] and the histogram of the probabilities in Figure 5.4.

The calibration in the range [0,0.01] in Figure 5.4 seems to imply that all models except the DT model are well calibrated in the range [0,0.005]. The predicted probabilities above 0.005 are all over-estimated, meaning the actual occurrence of claims is lower. The marker sizes already indicate that the majority of the predicted probabilities are in the range [0,0.003]. In the calibration plot of this range we see that the probabilities of the NN are under-estimated in these bins, more so than the LR model (line is further above the diagonal). As the majority of the predictions are located in these bins, the value of $\sum Pr(y|x)$ for the NN model is lower. The histogram of the probabilities show the same behaviour as many predicted probabilities of the NN are pushed towards zero. The histogram of probabilities also shows that only the DT model has predicted probabilities of exactly zero. The way probabilities for a DT model are calculated causes this behaviour as the probability for a claim will be zero if no claim records are located in a certain leaf node. The calibration in the region [0,0.003] shows that these low probabilities (and zero probability) are under-estimated for the true occurrence. In the higher probability regions the calibration of the DT model shows that those probabilities are overestimated. This behaviour results in the $\sum Pr(y|x)$ still being accurate. However this behaviour and the histogram of probabilities show that the probabilities resulting from a DT are not appropriate in a pricing model (we do not want to set a probability of zero for certain employees). The LR model shows well calibrated probabilities in the lower regions.

We showed that the NN model produces under-confident (probabilities are too low compared to actual occurences) probabilities. As we have a large number of non-claims this leads to the lower value for $\sum Pr(y|x)$. In Appendix D.2 we explain the choice of the hyper-

bolic tangent as the activation functions in the layers of the NN. This however, leads to a larger gradient descent if errors occur. This in turn increases the output of each neuron in forward propagation. If in the output nodes these model outputs are transformed using the logistic sigmoid this leads to probabilities closer to zero if the model is confident in a record belonging to the no-claim class.

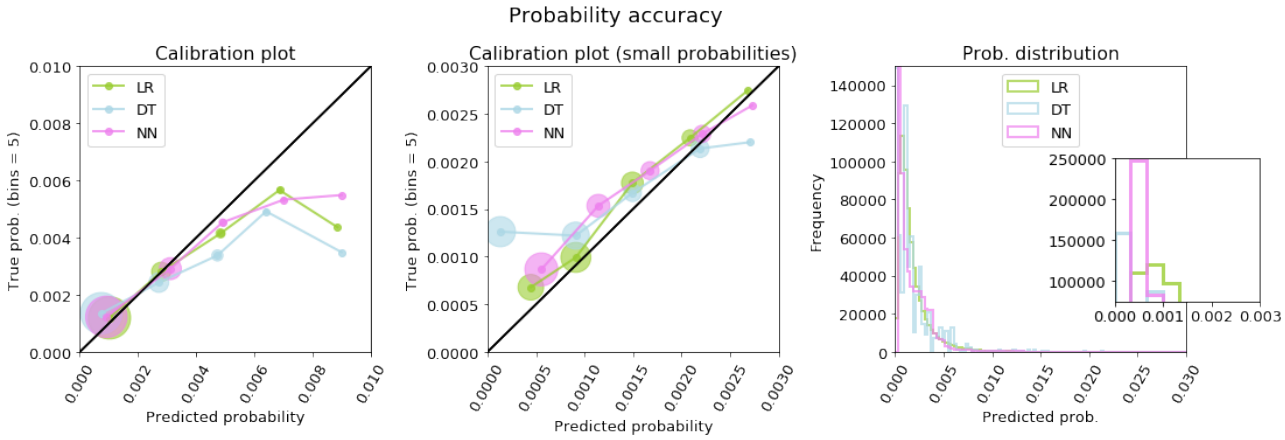

Figure 5.4: Calibration and prediction histogram of the DT, NN and LR models.

5.3 Data pre-processing for class imbalance

The previous section highlights that base classifiers, with optimized standard parameters, excluding techniques to handle class imbalance, can not achieve a high precision if we require a high recall. This results in a large percentage of non-claims being predicted incorrect. In this section we apply data pre-processing solutions to the class imbalance problem. We can incorporate these techniques in most model or ensemble structures and as such, do not consider it part of our initial data pre-processing stage described in Chapter 4.

5.3.1 Feature Selection

The first data pre-proccesing step to counter the class imbalance is by applying the GA algorithm presented in Section 4.5. We apply this for the different base classifiers described in Section 5.2 and "wrap" our GA algorithm around these classifiers. The base classifiers all use 232 features at the moment. The output of the feature selection process is presented in Table 5.2.

Table 5.2: Feature selection results.

Model	# features	μ AUC	orig. AUC	Δ AUC	μ BS	Δ BS	$\sum Pr(y=1\|x)$
LR	129	0.00972	0.00982	-0.00010	0.0018687	0.0000002	1296.5
DT	121	0.00885	0.00877	0.00008	0.0018743	0.0000028	1290.9
NN	130	0.00993	0.00987	0.00006	0.0018876	-0.0000173	1208.2

All models, except LR, improve as a consequence of feature selection in terms of class separation. The LR model needs many features in order to separate the data linearly. The DT and NN models seems to benefit from removing noisy features from the data. The DT model uses less features than the NN and LR model. NN models, often through non-linear combinations, can capture complex dependencies between features better. Hence, often NNs will benefit from more data and also features. However, we see that the BS improves for the LR and DT model but decreases for the NN model. Furthermore, the $\sum Pr(y = 1|x)$ of the NN model deviates more from the 1293 we expect. We already showed that the probabilities in the lower probability range of [0,0.003] of the NN model are underestimated. Using less features increases this effect. This can be expected as the class separation is improved. Hence, the model becomes more confident in its probability prediction for both classes. This causes the probabilities to be pushed closer towards zero due to the increased effect of the hyperbolic tangent activation function. Feature selection does heavily decreases the training time. This is especially true for the NN model. We opted to use the features selected by the GA algorithm for the remainder of the chapter for the models for which the AUC, and thus class separation, increased. This does cause worse probability estimates for the NN model. However, we argue that the decreased training time, improved class separation and the fact that the probability estimates for the NN were already not adequate without feature selection justify this reasoning.

5.3.2 Data re-balancing

Re-balancing data refers to adding minority samples or randomly sampling from the majority samples. We evaluate random under-sampling techniques where after we apply the over-sampling approach SMOTE.

Random under-sampling

Random under-sampling (RUS) is a simple approach in which we randomly sample data records x_n from the records with target $y_n \in \{0\}$ (zero represents no claim). A main drawback of using sampling techniques is the determination of the percentage in which we under- or over-sample. Hence, we assess the performance of the classification model g, in terms of the AUC of the PR-curve, over multiple values of s_{under} which we define as the under-sampling ratio. We use cross validation in which we randomly under-sample the training set in each fold. We test however on the 'normal' 5-fold test sets as this is the real data distribution we want to predict the probability for. We also plot the BS in order to assess the accuracy of the probabilities. In Figure 5.5 the mean AUC and BS for the base classifiers is shown.

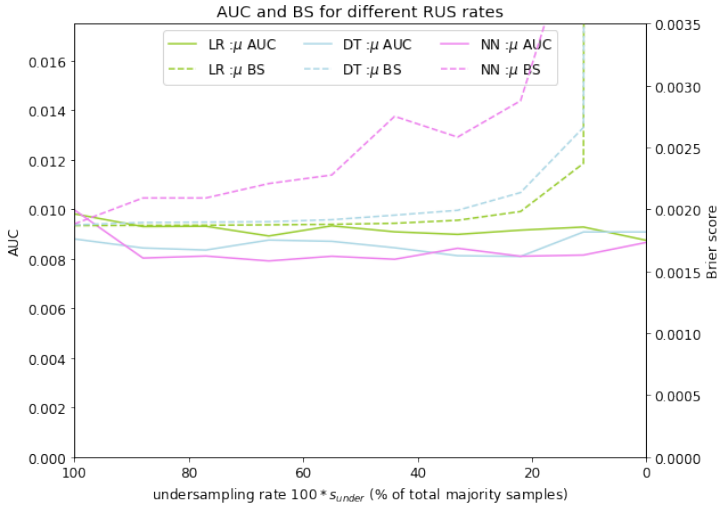

Figure 5.5: AUC and BS for different RUS rates.

The RUS sampling rates s_{under} range from 100%, so no under-sampling, to 0.018% and thus equal to the amount of minority samples. The AUC decreases for the NN and the LR model if s_{under} is decreased. Hence, discarding minority samples caused these models to have a worse prediction on the validation set. The AUC of the NN model decreases rapidly if s_{under} is decreased from one. This faster decrease can be attributed to NNs needing large amounts of data in order to update the weights efficiently. Under-sampling does improve the AUC of the DT model which is optimal when the class distribution in the training data is approximately balanced. However, we do see a steady increase of the BS, which leads to worse probability estimates. Concluding, we can say that RUS does not improve predictions for this dataset for most models. However, more importantly, the BS increases steadily if s_{under} is increases. Although some of this behavior can be attributed to worse probability predictions, the BS still increases if the AUC also increases for some s_{under} rates (i.e. the NN at $s_{under} = 0.7$). Hence, probability prediction get worse. Furthermore, note that when a very low s_{under} is applied the BS increases dramatically as we, deliberately, change the underlying distribution of the data we train the model on in a large way.

SMOTE

As mentioned in Chapter 3, SMOTE produces new “synthetic” minority samples based upon the distance between existing ones. As over-sampling method we apply SMOTE instead of randomly oversampling as this would include copying minority points, which might lead to over-fitting. This might be especially true in our very imbalanced data. We provide the standard SMOTE algorithm [56] in pseudo Algorithm 2.

Algorithm 2 SMOTE

INPUT: $X_{n,m}^{minority}, n \in N_{minority}, m \in M, \quad s_{over}, \quad k$ ▷ s_{over}, represents oversampling rate.

procedure SMOTE($X_{n,m}^{minority}, s_{over}$)
  for $n = 1$, $n{+}{+}$, while $n <= N_{minority}$ **do**
    Find k nearest neighbors of $X_{n,m}^{minority}$.
  Initialize P, with $P = N_{minority} * s_{over}$
  while $P > 0$ **do**
    Pick random number nn between 1 and k.
    for $m = 1$, $m{+}{+}$, while $m <= M$ **do**
      Determine distance $diff_m$ of $X_{n,m}$ to $X_{nn,m}$ ▷ pick random neighbor
      Pick random number gap_m between 0 and 1. ▷ Random data-point position
      New synthetic data-point: $X_{N+P,m} = X_{n,m} + gap_m * diff_m$.
    P = P - 1
  return $X_{n,m}, n \in N + P, m \in M$ ▷ Return same dataset with P new samples

We determine the number of minority samples and pick an sampling rate s_{over} Here s_{over} of two represent creating twice as many minority samples through SMOTE. We locate the k nearest neighbors of each minority samples (to other minority samples). We randomly pick a minority neighbor by picking a number between 1 and k. Then compute the distance between the nearest neighbor and the initial point for each feature m. We then create a synthetic data-point based on picking a random point between these distances for each feature m. In Figure 5.6 we visualize the new synthetic data points created using a principal component analysis (PCA) approach. We see here that using $s_{over} = 3$ creates new minority sample records in places between existing minority sample records.

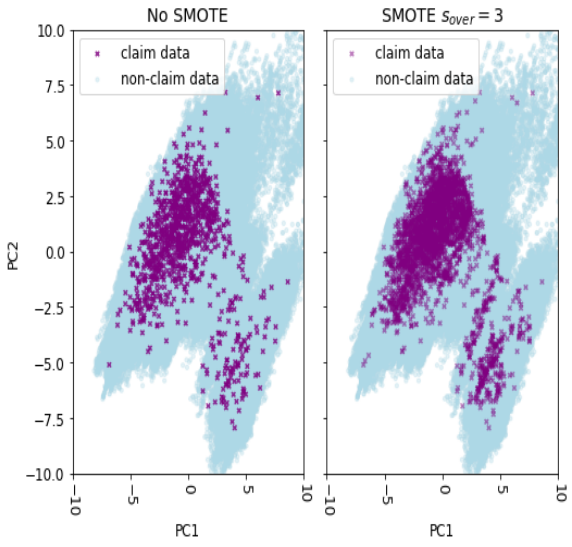

Figure 5.6: Illustration of SMOTE.

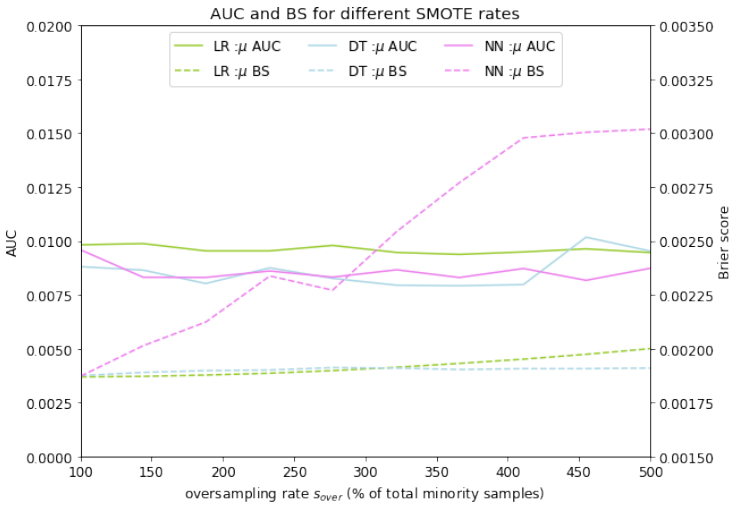

Figure 5.7: Performance of SMOTE.

In Figure 5.7 we apply the same method of finding optimal over sampling size s_{over} as for RUS. The DT improves due to oversampling, as well as the LR model. Applying SMOTE

on a NN severely increases the BS and decreases the AUC. The DT model seems rather sensitive to adding new samples. This can be attributed to the way outcome probabilities are computed. As for a DT, the probability is just the percentage of claims in the leaf nodes, training on added synthetic samples can alter the decision threshold taken in a significant way. Hence, when predicting on the validation set, the new thresholds can easily lead to very different predictions. Although some models have an improved AUC for some s_{over} rates, the same problem of using sampling methods persists here. The BS score increases for all models, although the increase is minimal for the DT model. Empirically determining the other metrics used for assessing probability accuracy also comply with the conclusion that sampling methods can decrease probability accuracy on imbalanced data.

Effect on loss function

The results of the data re-balancing techniques seem to indicate that, although the recall and precision can in some instances be increased, the predicted probabilities obtained are less accurate. The reasons for this can attributed to the effect of the class imbalance on the loss function used. We again use the toy datasets described in Sub-section 5.1 and plot the loss function for different class imbalanced data in Figure 5.8.

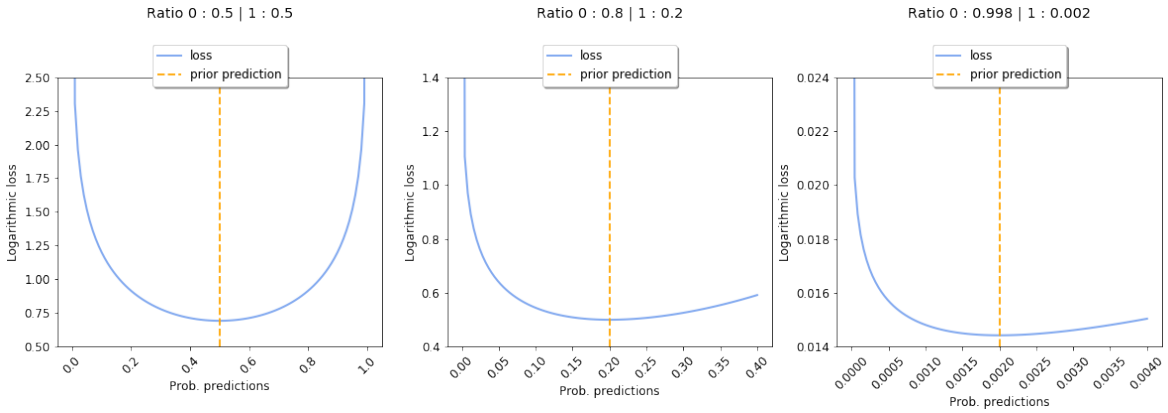

Figure 5.8: Effect of class imbalance on the binomial cross-entropy loss function.

We plot the logarithmic loss for predicting one equal probability for all data-points in the sets. With the prior prediction (set probability prediction equal to the percentage of minority samples) we show in which case the loss is lowest. For the balanced dataset this is intuitive. If we predict for all data-points 0.5 as a probability, the overall loss is minimal. The same reasoning applies to the other predictions we call prior probability (ratio of minority samples in the set). Although the predictions on our true dataset are not all equal, these loss function reactions give an insight into the inaccuracy of predicted probabilities after re-balancing.

By re-balancing the training data the algorithm will calculate the loss on this more balanced dataset. Hence for points that are harder to classify, the “save”, or less confident probability prediction will lie closer to the line of the prior prediction depicted in Figure 5.8. With under- and oversampling we are changing the prior probability in the training data.

Hence, $P(y = 1|s \neq 1) \neq P(y = 1|s = 1)$. Here s again refers to the over- or undersampling rates s_{over} and s_{under}. Due to this change, the posterior probabilities resulting from the models $P(y = 1|x)$ is also changed. As $P(y = 1|s \neq 1) > P(y = 1|s = 1)$, [15] $P(y = 1|x, s \neq 1)$ will generally be higher than $P(y = 1|x, s = 1)$. Hence, re-balancing leads to higher average predicted probabilities, which in turn leads to worse values of the BS.

Concluding we can say that data re-balancing methods in highly imbalanced datasets in order to predict probabilities lead to inaccurate probability predictions. Cost-sensitive methods such as cost weighting have similar properties and proved to have the same effect on probability predictions.

Transformation of probabilities after re-balancing

The resulting probability estimates from machine learning algorithms are often used as a ranking on which a threshold is to be set. However, we showed that using these resulting probabilities directly leads to inaccurate probabilities. This was caused by a change in the prior probability in the training data. In literature not much emphasis is placed on the transformation of probabilities after re-balancing. [57] proposes a method to transform the probability estimates when a model is applied on a test set, for which the class probability differs from the training set. The same problem is essentially what is occurring if one re-balances the training data using over- and undersampling and testing on the normal data. Hence, we apply the method proposed in [57] to transform our probability estimates.

By applying Bayes' theorem we have for our normal (test) data that,

$$P(x|y = 1) = \frac{P(y = 1|x)P(x)}{P(y = 1)}. \tag{5.5}$$

Equation 5.6 also applies to our under- or oversampled training data for which we denote probabilities by P_s. We want the adjusted posterior probability $P(y = 1|x)$, based on the posterior probability estimate resulting from model G. Model G is trained on the resampled training data x_s and thus we denote the, from model G resulting, uncorrected probability, as $P_s(y = 1|x)$. From Equation 5.6 we have that,

$$P(y = 1|x) = \frac{P(x|y = 1) * P(y = 1)}{P(x)}. \tag{5.6}$$

Based on the assumption that $P(x|y = 1) = P_s(x|y = 1)$ [16] and by setting a term $f(x)$ as $\frac{P_s(x_s)}{P(x)}$ we get,

$$P(y = 1|x) = f(x) * \frac{P(y = 1)}{P_s(y = 1)} * P_s(y = 1|x). \tag{5.7}$$

[15] Both with over- and undersampling the prior probability of the claim class increases.

[16] Note that this is a rather harsh assumption. We explain this in Appendix C.3.

From Equation 5.7 we can get cf., [57]

$$P(y=1|x) = \frac{\frac{P(y=1)}{P_s(y=1)} * P_s(y=1|x_s)}{\sum_{j=1}^{2} \frac{P(y=j)}{P_s(y=j)} * P_s(y=j|x_s)}. \tag{5.8}$$

Equation 5.8 (see Appendix C.3 for derivation) can be used in order to transform probabilities $P_s(y=1|x_s)$ resulting from training model G on re-balanced data x_s. The terms $P(y=j)$ refer to the ratio of samples of class j (non-claim or claim) in the original data and $P_s(y=j)$ to the ratio in the resampled training data. Note that in the case of undersampling the transformed probabilities for the claim class, $P(y=1|x)$, are larger than the ones resulting from the model and smaller with oversampling. The assumption $P(x|y=1) = P_s(x|y=1)$ is violated as with using sampling methods we are either removing records x with undersampling and adding synthetic records based on previous records x in case of oversampling. For instance, when applying a low s_{under} value we are removing a lot of records, which will lead to the distribution of x to be different in the re-balancing training set from the test set. In spite of the harsh assumption we review the effect of this transformation by plotting the SMOTE results in Figure 5.9.[17]

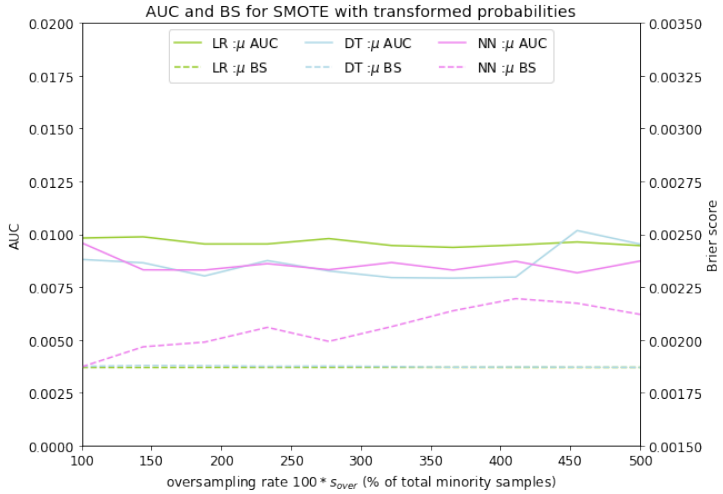

Figure 5.9: AUC and BS for SMOTE with transformed probabilities.

The BS of all models improves when applying the transformation. The BS of the NN increases due to the oversampling although in a far less extreme manner. As expected, the AUC of the models remain constant. However, we transform the resulting probabilities directly. Although we see an increase in AUC for the DT model and a relatively constant BS, the sum over the resulting probabilities for the DT model is too low. The probabilities resulting from the DT model are often zero (see Sub-section 5.2.4). By transforming the probabilities directly the non-zero probabilities are lowered in case of oversampling but the probabilities that were zero do not. Hence, although the BS remains low for the DT model,

[17] We show the results of SMOTE as oversampling worked better than undersampling. Furthermore we note that this method does not require retraining a particular model and can be applied directly on the predicted probabilities.

the transformation results in inaccurate probabilities. For undersampling the probabilities that are not zero are increased slightly, although we note that this effect is less extreme than is the case with oversampling. The LR model does seem to produce accurate probabilities with the transformation. However, the sum over probabilities decreases slightly in case of SMOTE. The reason this is the case, especially for higher values of s_{over} is that the assumption, $P(x|y = 1) = P_s(x|y = 1)$, made in the probability transformation is violated. Hence, by applying the transformation proposed by [57] to the probabilities after data re-balancing techniques we can increase the class separation while retaining the accuracy of probabilities. However, besides the BS, other probability evaluation metrics should also be considered in order to see if these are still correct. Based on these other metrics, we conclude one should not apply this method on the probabilities of a DT model, or other models, in which probabilities result in zero before the transformation. Furthermore, the transformation can still leads to inaccurate probabilities if s_{under} is low or s_{over} is high.

5.4 Algorithmic solutions to the class imbalance

In Subsection 3.2.2 we outline different algorithmic solutions to the class imbalance problem. The best performing solution in practice and literature are ensemble based. The primary ensemble methods are Bagging, Boosting and Stacked Generalization.

We do not consider Stacked generalization due to the lack of interpretability. The use of a meta-classifier in order to combine predictions of multiple other classifiers is (too) complex to be incorporated as a pricing model. Bagging allows for a reduction in variance in performance on a test set. However, as bagging is a averaging method over multiple base classifiers, predicted probabilities tend towards the prior probability instead of moving towards zero and one [58]. Furthermore, bagging can reduces the variance, however, the bias of the model will not be reduced. Some of the weak learners might capture records which are hard to classify (minority samples). However, the results of these weak learners is adjusted through other predictions. Boosting is a form of ensemble learning in which the variance and bias can be reduced. Hence, in this research we focus on boosting algorithms as algorithmic solution to the class imbalance problem.
With boosting, the algorithm fits weak classifiers sequentially to improve a combined 'strong' classifier. We apply the Adaboost (AdaB) and (logarithmic loss) Gradient boosting (GB) algorithm. In order to get a deep understanding of the boosting models and be able to adjust all parameters we programmed the algorithms ourself in *python*.

5.4.1 AdaBoost

In the first, well-received, boosting algorithm AdaB, the boosting model has the form,

$$G_T(x) = \sum_{\tau=1}^{T} \alpha_\tau g_\tau(x; w_\tau)$$

Here, $g_\tau(x; w_\tau) \in \{-1, 1\}$, are "weak classifiers" applied in iteration τ. Often a "weak classifier" is used, referring to a slightly above random performing classifier, as it has a high variance. Hence, in each iteration new records can be classified correctly. The parameter a_τ is included to give more gravity to weak learning with a low error. The weak learners are applied to training data vectors $x_n, n \in N$ and target $y_n \in \{-1, 1\}$. We adjust the outputted prediction $\hat{y}_n$ as we want to transform the output of $G_T(x)$ using the sign for classification [18]. The main idea of the AdaBoost algorithm is to put more weight, represented by w_τ, on training data which is hard to classify. The error obtained by weak classifier g_τ is [48],

$$E_\tau = \sum_{n=1}^{N} w_{n,\tau} I(y_n \neq g_\tau(x_n)).$$

Here $w_{n,\tau}$ resembles the weight of training data record n in iteration τ. The parameter a_τ is then obtained by,

$$a_\tau = \frac{1}{2} log(\frac{1 - E_\tau}{E_\tau}).$$

The next weak learner $g_{\tau+1}(x; w_{\tau+1})$ will be trained using weights,

$$w_{\tau+1,n} = \frac{w_{\tau,n} * e^{-a_\tau y_n g_\tau}}{Z_\tau}. \tag{5.9}$$

Here, Z_τ, is a normalization and is equal to the sum over n of the nominator of Equation 5.9. We initialize weights w_0 with $1/N$. To find a binomial prediction one takes the sign of $G_T(x)$. To get the the conditional probability, $P(y = 1|x)$, of a record belonging to claimed class we use the sigmoid function as we have a binary target and the outcome $G_T(x)$ can be seen as a linear combination with weights a_T. The reasoning behind the algorithm, and its equations, are outline in Appendix E.

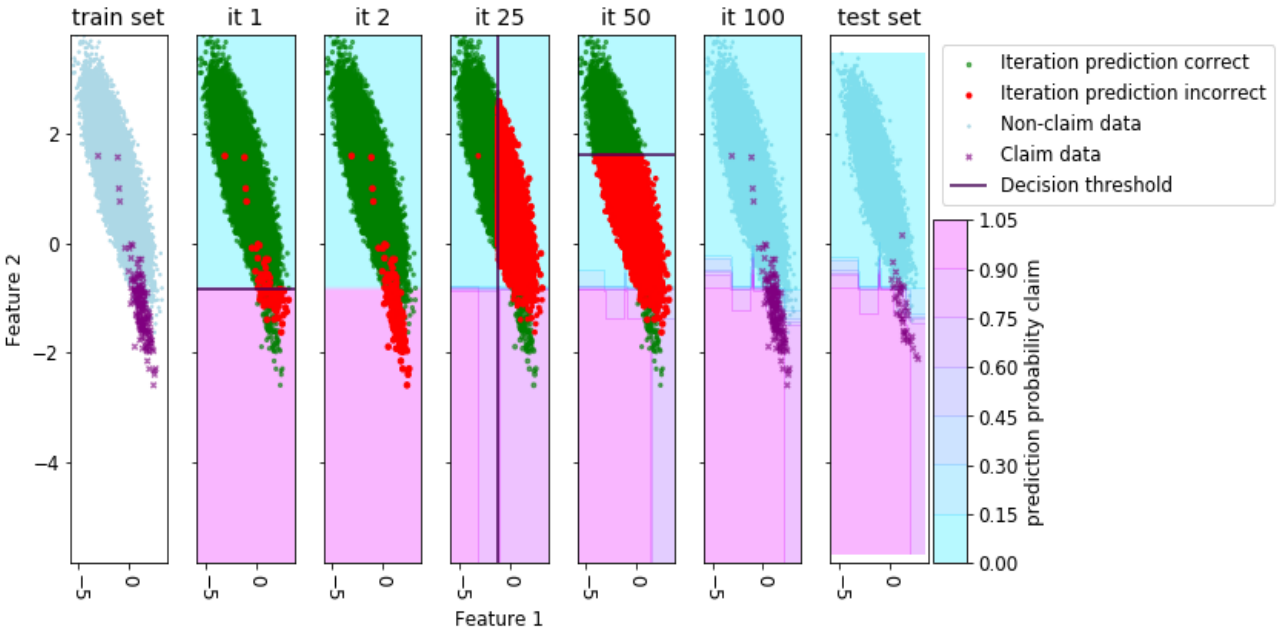

Figure 5.10: Adaboosting iterations

[18] Note that other implementation of the AdaB algorithm do not convert the target values y_n.

In Figure 5.10 iterations of the AdaBoost algorithm are visualized.[19] We represent the distribution of the claim and non claim data in the training set representing 80% of the toy data. We see that in the first iteration the boundary is placed to predict a large section of claim data correctly. However, this also leads to many falsely predicted non-claim data. The AdaBoost algorithm notices this and sets a threshold leading to all predicted as zero in iteration 2. Note, that in iteration 2 the probabilities do not change a lot. This is due to a much smaller value of a_2 compared to a_1. The predicted probability of a claim is then adjusted each iteration until we get a more "smoothed" predicted probability which is above 0.5 around the claim data points and to the south-west of the claims. In iterations 50, we can see that the threshold is also set around the 'harder' claim records. However, as due to this threshold many other points are incorrectly classified the a_{50} value is again low. This is however positive as when the trained model is applied to the test set we again see a high probability of a claim around where the true claim data is located. The 'noisy' claims records in the training set did not lead to wrongly predicted non-claim data on the test set. Note that we use train on two data for two features. However, the process of the AdaBoost algorithm is the same in the higher dimensions of our true data. Through this process we can see the main drawback of using a single tree structure. This can not achieve overlapping of multiple regions. We can clearly see here that Adaboost algorithm can accomplish this.

5.4.2 Gradient boosting

Several years after the AdaBoost algorithm, explained in Subsection 5.4.1, was introduced, [42] developed a statistical framework for boosting in general. They showed that AdaB could be seen as "stagewise, additive modeling" using the exponential loss function. With other loss functions and applying a, newly introduced, GB algorithm, one could often obtain better results. The reasoning is to apply a form of gradient descent to reduce the error. In the, most practiced, case of DT GB, we induce a regression tree [20] $g(x;\theta_\tau)$ in iteration τ. This weak learner g is trained to predict the negative gradient [48]. The reason for this is to iteratively reduce the loss function. Assume we initialize a model $g_0(x;\theta_0)$ and predict for x. We have,

$$g_0(x;\theta_0) = \hat{y}.$$

We look at the loss obtained by a loss function $L(\hat{y}, y; \theta_\tau)$. The loss obtained represents how far the prediction $\hat{y}$ is off. Now we would want to improve the model by adding another regression model $g_1(x;\theta_1)$ in order to have a prediction equal to target y. Or to have that,

$$g_0(x;\theta_0) + g_1(x;\theta_1) = y.$$

Which we can rewrite as,

$$g_1(x;\theta_1) = y - g_0(x;\theta_0).$$

[19] The visualization and decision boundary drawing is applied to two features on a toy dataset. The set has an class imbalance of 99,8% negatives versus 0,2% positives.

[20] By predicting on a continues value we need a regression model.

Hence, to reduce the loss in the next prediction, we do not train g_1 on target y, but on the $y - g_0(x; \theta_0)$. We can interpret this as the negative gradient of $L(\hat{y}, y; \theta_0)$. We explain the use of a negative gradient in Appendix F.

In our case we use the same binary logarithmic loss as defined in Equation 5.2. It can be shown (see Appendix C.2) that the negative gradient of $L_{ce}(\hat{y}, y; \theta_\tau)$ corresponds to,

$$\frac{\partial L(\hat{y}, y; \theta_\tau)}{\partial \hat{y}} = y - \sigma(\hat{y}).$$

The GB model we apply is of the form,

$$\hat{y}_T = \sum_{\tau=1}^{T} g_\tau(x; \theta_\tau) = g_T(x; \theta_T) + v * \hat{y}_{T-1}.$$

Here $\hat{y}_{T-1}$, is the prediction of the last iteration. As we are learning on the negative gradient, $\hat{y}_{T-1}$ can be seen as a gradient descent direction. Parameter v is the step-size we take.

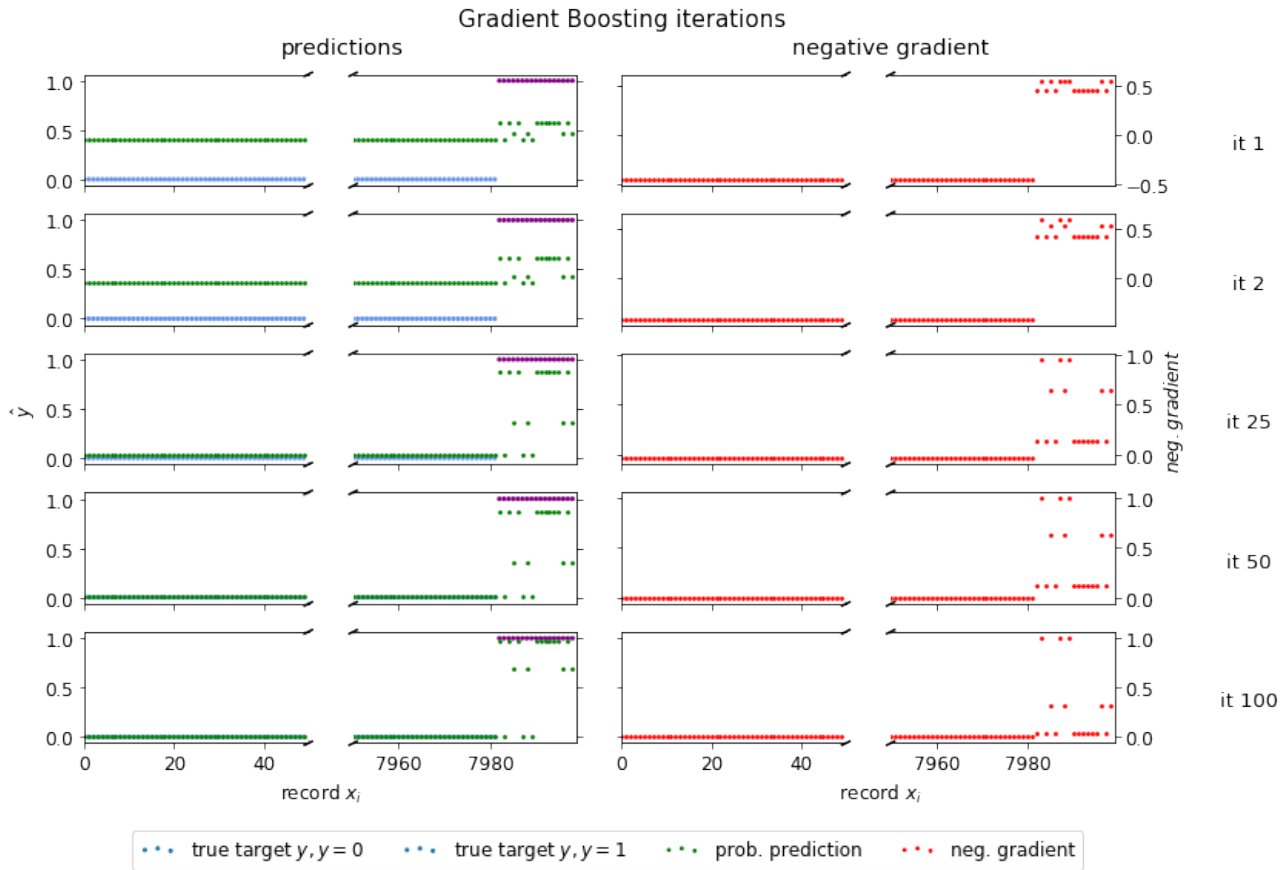

Figure 5.11: Gradient boosting iterations.

In Figure 5.11 we visualize the GB procedure. We again produce a toy dataset with a class imbalance resembling the true data distribution. We use 10.000 data-points from which we take 80% as training data. In the plot we do not show a large part of the records of the minority class for visibility reasons. Through a regression tree which is trained over

the negative gradients, the next negative gradient is reduced each iteration leading to the predicted probabilities $\hat{y}$ getting closer to the true class labels. If the gradient over all points converges to zero we obtain the optimal solution. As with the Adaboost algorithm, the algorithm has a problem with predicting some 'harder' records put in the toy dataset. For the GB algorithm the resulting predicted probability is also 'smoothed' due to different threshold set by the individual trees (similar to Figure 5.10). Eventually, these individual threshold result in a probability prediction on the test set.

5.4.3 Comparing boosting algorithms

In Figure 6.1 we plot the PR-curves of the AdaB and GB algorithms. Again we see that achieving a high precision can not be achieved by both models. However, the performance is increased with respect to the base classifiers and lies above a random model. The GB algorithm performs better as it can reach higher precision scores in the lower range of the recall. The AUC of the GB is better then the, till now in terms of class separation, best model which was the NN. The performance of the AdaB algorithm is in line with the NN model. Note that the precision for the boosting models is slightly higher if we require a recall of 0.3.

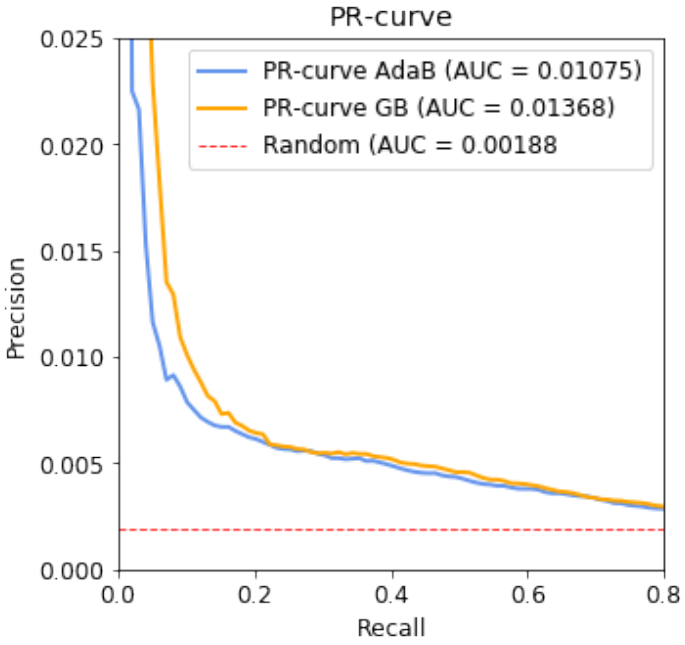

Figure 5.12: PR-curves of Adaboost and Gradient boost.

With boosting algorithms one important setting is the number of iterations to perform. In Figure 5.13 we evaluate performance over the iterations.

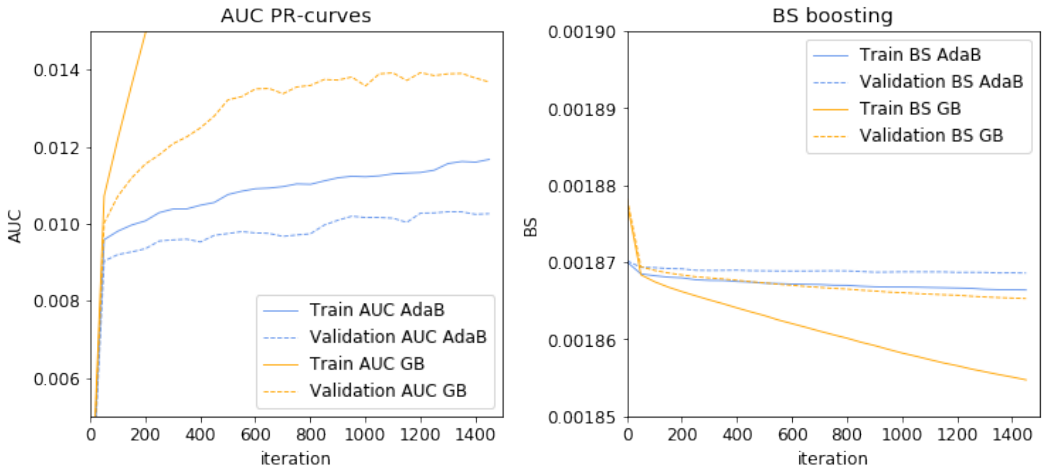

Figure 5.13: AUC and BS of Adaboost and Gradient boost.

The AUC of the GB algorithm increases to higher values in considerably fewer iterations than the AdaB model. We only plot till iteration 1500 in this plot. The AUC of the GB model on the validation set decreases after around 1100 iterations. The learning rate v of the GB model in Figure 5.13 is set at 0.10. With $v < 0.10$ the incremental increase was very slow, if we increase v, we saw decrease in performance on the validation set as the model seems to overshoot the local/global minima. We set the number of iterations of the GB model to 1100. The number of iterations for the AdaB model is set to 2000 to increase performance. We describe the choice of parameter setting for the AdaB and GB models in respectively Appendix D.3 and D.4. The BS of the GB algorithm decreases more, and faster. The BS of AdaB decreases incremental over the different iterations. The BS of the GB method and AdaB model is lower than the BS of the LR model.

Although the BS of the boosting models decreases to low values, we still assess the accuracy of probability predictions using our other metrics. Figure 5.13 already shows the BS of both models. The sum over the estimated probabilities is 1223.0 for the AdaB model and 1291.1 for GB. The amount of claims in the set corresponds to 1293.

The results of the AdaB algorithm suggest that the probability predictions are not very accurate although the BS is low. In order to visualize the calibration of the probabilities we plot the calibration plot and a histogram of predicted probabilities in Figure 5.14. Again we mention that we the vast majority of the resulting probabilities are lower than one percent. Hence, we only show the calibration up until this point. We also plot the calibration in the region [0,0.003] again. We see that the predicted probabilities higher than 0.006 are overestimated for both models, as was the case with the base classifiers. The observed frequency in those bins is lower. In the probability regions below 0.006 the calibration is good for both models. However, the low value of $\sum_{n=1}^{N} P(y = 1|x_n)$ of the AdaB algorithm implies that the AdaB probabilities are not calibrated well. In the histogram of probabilities, visualized in Figure 5.14, we see that most probabilities for the AdaB model have been “pushed” towards zero. We zoom in on the lowest two probability bins (bin size = 0.0005). We see

that the number of predicted probabilities in the lowest bin is higher for the AdaB algorithm. In the calibration plot of the lower probabilities in the range [0,0.003] it can be seen that AdaB underestimates the probabilities in these bins. As many records are concentrated in those probability bins, underestimation here results in a higher BS and a too low value of $\sum_{n=1}^{N} P(y = 1|x_n)$. Furthermore, this leads to the large deviation in summed probabilities (1293 claims are in the training/validation set). Hence, the predicted probabilities of the AdaB model are (more) often close to zero. The model is very confident on the prediction that these records will not result in a claim. However, the sum over the probabilities show us that this is not in line with the expected probability.

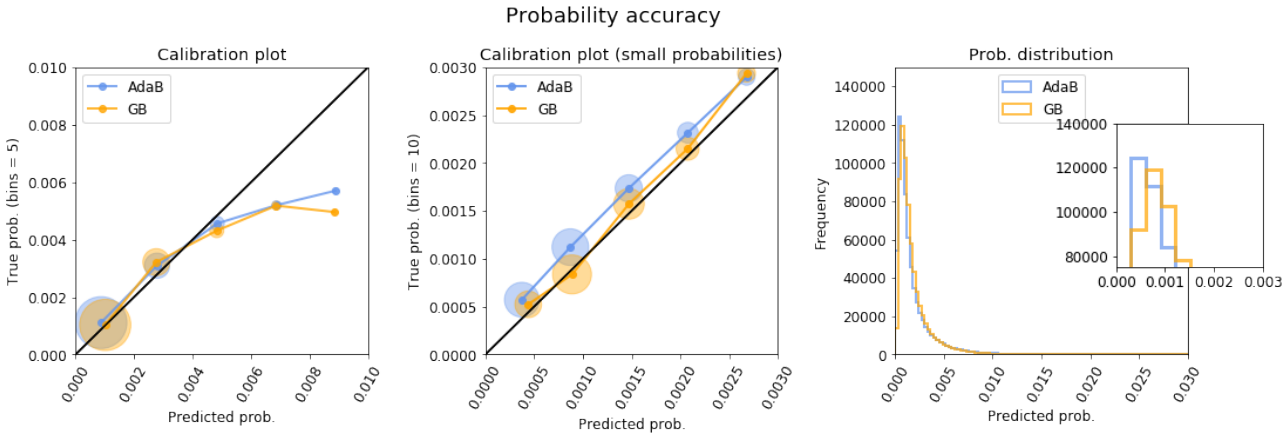

Figure 5.14: Predicted probability accuracy of the boosting algorithms.

The results of the AdaB algorithm, which minimizes the exponential loss function, produces worse probability estimates. In Figure 5.15 we plot different loss functions for a case where the true target label is class 1. We adapt the logarithmic loss of the GB algorithm by setting the target labels to $y \in \{-1, 1\}$.

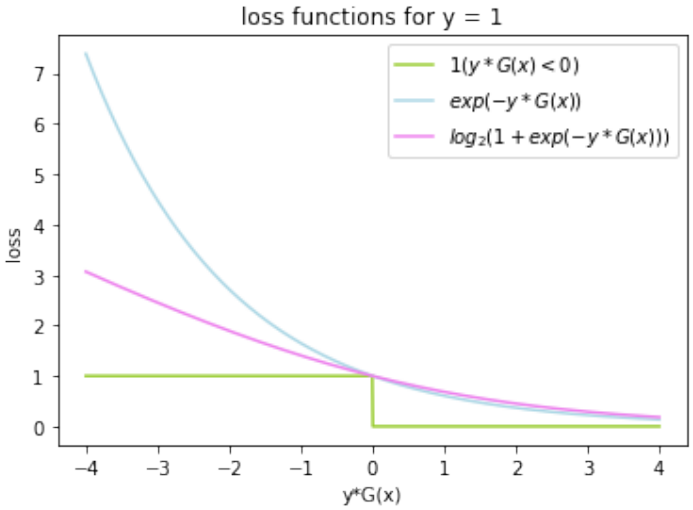

Figure 5.15: Boosting loss functions [26].

We add the simple misclassification loss, which is $\mathbb{1}(y * G(x) < 0)$, for interpretation. As y represents our true label and $G(x)$ the result of the algorithm [21] a negative value of $y * G(x)$ represent a wrong prediction. If $G(x) < 0$, which, by for instance taking the sign in Adaboost, suggest a prediction for the class -1, then if true label $y = -1$ we get a positive value and thus a correct prediction. Otherwise if $y = 1$ we get a negative value. Thus for the simple misclassification cost we would get a loss of one if incorrect and zero if correct. The degree of $y * G(x)$ represent the confidence in the prediction as outputted by the model.

As the same intuition applies to the exponential and binomial logarithmic loss function we see that the main difference is in how "hard" wrong prediction are penalized. AdaBs exponential loss function penalizes wrong predictions in exponentially increasing order. The binomial logarithmic loss does this in a more linear fashion. In theory this should lead the AdaB algorithm to capture harder to classify border points faster. The wrongly classified, often minority class, records are penalized hard and the the next iteration more emphasis is placed on these points. This is normally an advantage of the AdaB algorithm in terms of classification of minority samples in a imbalanced data setting. On our dataset, the algorithm seems incapable of capturing many of these harder points (the probability prediction of minority points, that can be captured, are pushed toward zero very fast). However, in contrary this same behavior negatively affects the accuracy of the probability predictions which are pushed towards zero and one.

The reason for this is that $|G_T(x)|$ produced by AdaB *and* GB increases in most iterations. An intuition for this can be seen in Figure 5.11 where the negative gradient on which the next tree is trained will produce prediction $\hat{y_\tau}$ being negative or positive based on the value of the negative gradient. Thus by producing new trees the score of $|G_T(x)|$ increases. For classification purposes in which the sign is taken this is irrelevant. However, as more iterations are added, the probabilities are pushed towards zero and one. The same holds for AdaB where g_τ is either positive or negative and thus $|G_T(x)|$ increases with scaler a_τ. The reason why AdaB with is exponential loss function produces worse probability estimates can attributed to the negative gradient of the loss function. In the algorithm a_τ represent the gradient step, v in the GB algorithm, taken when applying boosting on the exponential loss function (see Appendix E). By expecting Figure 5.15, the gradient step produced by the exponential loss function will be far steeper if predictions are even slightly wrong. Thus, if the model adapts to predict some 'harder' to classify minority samples and thereby also (wrongly) increases the probability of majority samples, the loss on these slightly wrong predictions of majority samples result in a higher loss as would be the case for the logarithmic loss function. Furthermore, as we assume the classes given by y are true in the data, possible records with employees who might have a high chance of becoming disabled to work, will also have probability prediction close to zero. This due to the models assumption that this records should have a prediction of zero. The AdaBoost algorithm punishes errors on these possible high risk employees that much more.

[21] Note, that almost all machine learning algorithms do not produce $G(x)$ directly. They produce a value for each x which is later transformed, using for instance the logistic sigmoid function, to probability estimate $G(x)$.

5.5 Test set performance

We have tuned all parameters of the different models while testing on the validation sets using K-fold cross validation. The next step is to compare the performance of these models with the parameters found on the test data. In this case we train our models on the entire training set (no more cross validation). We test, and find our model performance, on the test set as described in Section 4.4. The results are described in Table 5.3. We show the AUC, the BS and the sum of the probabilities. We show these metrics for each model with the different methods applied in order to handle the class imbalance (feature selection, RUS and SMOTE). Furthermore, we present important parameters used per model in a italic formatted section per model. The parameters listed apply to the specific method used. For instance, the s_{under} parameter applies to the s_{under} rate applied in the RUS method for a model. The parameters under the feature selection methods refer to the number of features used. These parameters are based on results on the training/validation set described in this chapter. The parameters listed under the "Normal" section apply to important parameters used for the different models for each of the methods (for each different NN model we use a network size of (20,20,20) and the hyperbolic tangent activation function).

model	metric	Normal	Feat. selec	RUS	SMOTE
	parameters		*# = 129*	$s_{under} = 0.67$	$s_{over} = 2.77$
LR	AUC	0.00428	0.00427	0.00414	0.00443
	BS	0.0018739	0.0018736	0.0018744	0.0018740
	$\sum P(y = 1\|x)$	144.0	144.2	143.6	143.8
	parameters	*depth = 7*	*# = 121*	$s_{under} = 0.11$	$s_{over} = 4.55$
DT	AUC	0.00410	0.00630	0.00315	0.00415
	BS	0.0018792	0.0018709	0.0018799	0.0018766
	$\sum P(y = 1\|x)$	143.9	144.7	144.7	77.6
	parameters	*size = (20,20,20), tanh*	*# = 130*	$s_{under} = 0.33$	$s_{over} = 4.11$
NN	AUC	0.00475	0.00520	0.00330	0.00466
	BS	0.0018762	0.0018744	0.0027846	0.0019654
	$\sum P(y = 1\|x)$	156.3	76.9	245.8	160.6
	parameters	$\tau = 2000$	*# = 119*		
AdaB	AUC	0.00600	0.00616	-	-
	BS	0.0018723	0.0018730	-	-
	$\sum P(y = 1\|x)$	137.2	139.0	-	-
	parameters	$\tau = 1100, v = 0.10$	*# 120*		$s_{over} = 3.00$
GB	AUC	0.02292	0.02255	-	0.00526
	BS	0.0018609	0.0018627	-	0.0018741
	$\sum P(y = 1\|x)$	143.6	143.6	-	72.7

Table 5.3: Performance of models on test set.

The ranking of different models based on performance is what we would expect from the training set but the actual AUC is lower for most models. By inspecting the PR-curves this is mostly due to the lower precision for low levels of the recall. On the test set most models have more difficulty separating a few number of claims from non claim data. Similar to results on the training set, the GB model proves to obtain the best class separation based on the AUC. By inspecting the PR-curve, the GB model achieves a relatively high precision for the low recall levels. The second best AUC of different models without techniques to handle class imbalance is the AdaB model. The vital difference in performance of these boosting models is the probability accuracy. The AdaB model has a relatively low BS, however the sum over the probabilities diverges much from what is to be expected. For the test data this should sum up to 143. The same holds for the NN models. The GB model performs quite well in terms of probability accuracy. The BS is low and and sum over the resulting probabilities is close to the actual occurrences. With respect to data re-sampling methods we again transform the probabilities using the method explained in Sub-Section 5.3.2. The LR model with a SMOTE approach increases performance and the transformed outputted probabilities are representative of the true data. We again note that using transformed probabilities for the DT model leads to inaccuracies as many predicted probabilities resulting from the model are zero. We also apply SMOTE for the GB model. The s_{over} rate is again based on results on the training/validation set. The AUC on the validation set also decreased due to oversampling while the training set accuracy increased. Hence, with this oversampling rate the boosting method led to overfitting. As boosting models focus on harder to classify points, which are often the minority samples, over-sampling leads to focus on these new synthetic records. Thus boosting in general is susceptible to overfitting while using sampling methods.[22] Furthermore, these results show that the transformed probabilities result in inaccurate probabilities. The feature selection using the GA algorithm also improved the performance, both in class separation and probabilistic accuracy for most models. This result shows that many features can be discarded while still obtaining good class separation. We do note that applying the GA algorithm on the boosting models was time consuming. The performance of the GB model did not improve due to the GA algorithm.

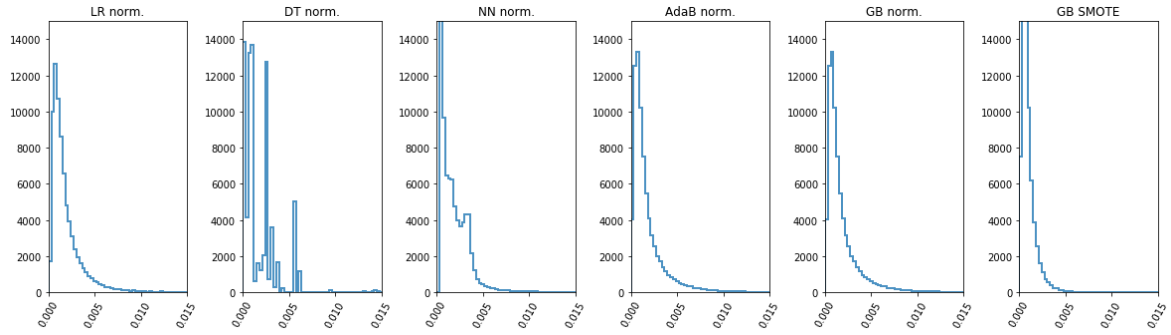

Figure 5.16: Predicted probability distributions on the test set.

[22] Due to the decrease in performance on oversampling while using the GB model, we opted not to apply other sampling methods on the boosting models.

For a selection of models we plot the distribution of the predicted probabilities in Figure 5.16 to assess whether the distribution is in line with what we would expect. The predicted probabilities of the NN models again show that probabilities are pushed towards zero while the DT model displays a “non smooth” probability distribution. The GB model is much more in line with the LR model as is the AdaB model. The LR model directly optimizes the binomial log loss function. This leads to accurate probabilities in general. However, the LR model can not achieve a good class separability. The GB, which uses GB to optimize the log loss function, produces probability estimates resembling the shape of the LR model. Although the probabilities of the AdaB also resemble the LR model, the summed probabilities show that indeed still to much probabilities are pushed towards zero. The same holds for the corrected probabilities for the SMOTE technique applied for the GB model. In light of these results, the model we select is the GB model using 1100 iterations, a learning rate v of 0.10 and using all features.

Chapter 6

Results

We have produced a model which has proven to produce adequate class separation and provide prediction probabilities with good results on the test set. One of the research goals was to see whether, by using more advanced models and internal/external data, it is possible to improve the in-flow probability estimates. We therefore check the improvement over the current model in Section 6.1. Furthermore, it is important to examine the influence of different features on the outcome of the model. We review feature importance in Section 6.2. In Section 6.3 we investigate how the model responds to changes in the feature values. In Section 6.4 we examine the probabilistic accuracy. Throughout the chapter we visualize parts of dashboards created in order to investigate the produced model. Finally, in Section 6.5 we outline the implementation of the entire process described in this research.

6.1 Comparison with the current model

In order to assess the improved probability prediction of the GB model we compare the performance with the current pricing model. As mentioned in Chapter 2, the current model differentiates on age and salary.[23] The GB uses a considerable larger number of features. Below we plot the PR-curve for the current and GB model.

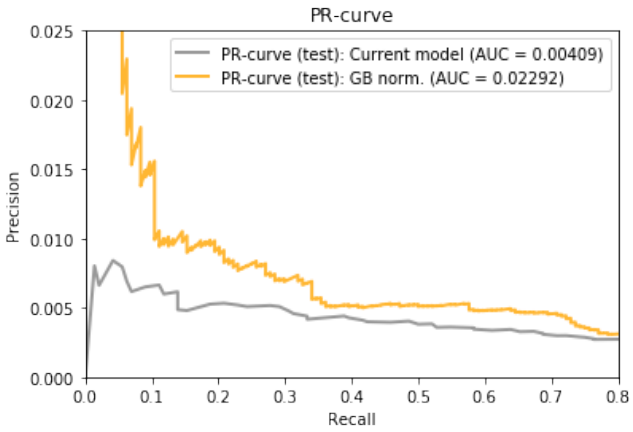

Figure 6.1: PR-curves of Gradient boost and current model.

[23] We will not discuss the exact parameters of the current model due to the sensitivity of this information.

The AUC of the current model is low compared to the GB model, but in line with some of the worst performing models on the test set. As the current model only uses two features this shows that improvements are hard to make in the imbalanced data setting. The increase in the AUC of the GB model might implicate a very small improvement. However, a 0.001 higher precision at for instance a recall level of 0.6 imply approximately 50000 fewer non-claims being predicted as a claim for that particular threshold. As we are dealing with many employees these slight difference in separation between classes have a large impact in terms of pricing. The sum over the probabilities for the current model on the test set is 153.0 with a BS of 0.0018735. Hence, the BS of the GB model is 0.0000126 lower. So, on average over all 76650 records in the test set, the difference between the probability of occurrence for one record (as we know whether a claim occurred we set this as zero or one) and the probability output of the model is 0.0000126 less in the new model.

6.1.1 A realistic pricing model

We have showed that adding more features to the pricing model in combination with more advanced machine learning techniques can improve class separation. However, many features might not be appropriate in converting this model into a pricing model. For instance, although the feature importance is low, the feature "K_WON_OPP_DEELNR" representing an indication of the square living area of housing in the area is not something we necessarily want to base our estimates on. In order to check whether the model still produces adequate predictions while using far fewer features, we again train the model using only 26 features of the 73 before. We implement some geographical features of the company and employee, the yearly salary, age and the amount of time an employee has been in the contract.[24] With these 26 features the AUC of the model drops to 0.02076. Hence, due to the decrease in number of features, a number more non-claims are predicted as a claim. However, the AUC still implies a large improvement over the current model. The BS and sum over probabilities are respectively 0.00186624 and 142.5 (in the test set there are 143 claims) for the new GB model with 26 features. Hence, probability accuracy does not decrease substantially either. Hence, even with a large decrease in the number of features and retaining only plausible features for a pricing model we still obtain adequate results. For the remainder of this chapter we will apply the results from the GB model with less features to give more insights into the importance of some of these particular features. [25]

6.2 Feature importance

A first indication of the importance and influence of different features on the model outcome is by calculating feature importance. We calculate the feature importance using the reduction

[24] We also add some features based on these features. We especially are concerned with features that seem plausible in a pricing model.

[25] Note that most of the chosen features were among the most important in the GB model using all the features. We selected some of these features with these feature importances in mind.

of the criterion caused by that feature. In our case, this criterion is the mean squared error of the individual regression trees in the GB model. Remember, we train on the negative gradient, as such we need a regression type impurity measure. For a single tree T the feature importance is calculated through the Gini-impurity in each node j [59]. This is,

$$I_{tree} = \frac{N_j}{N} c_j - \frac{N_{j,left}}{N} c_{j,left} - \frac{N_{j,right}}{N} c_{j,right}.$$

Here N_j refers to the number of records in the node j and its left and right children, N to the total number of records and c refers to the Gini impurity. Note that in our boosting algorithm we apply a tree depth of one. Thus $\frac{N_j}{N} c_j$ corresponds to 0.5 (highest possible Gini impurity value), with j as our (only) decision node. Hence, we have a value of 0.5 if the Gini impurity is small (thus good) in the children nodes. Within our boosting algorithm, we have multiple additive trees. To get the total feature importance we average over the different trees to get the feature importance of the boosting model in total [26]. Thereafter we compute the importance of each feature relative to the other features. [26]

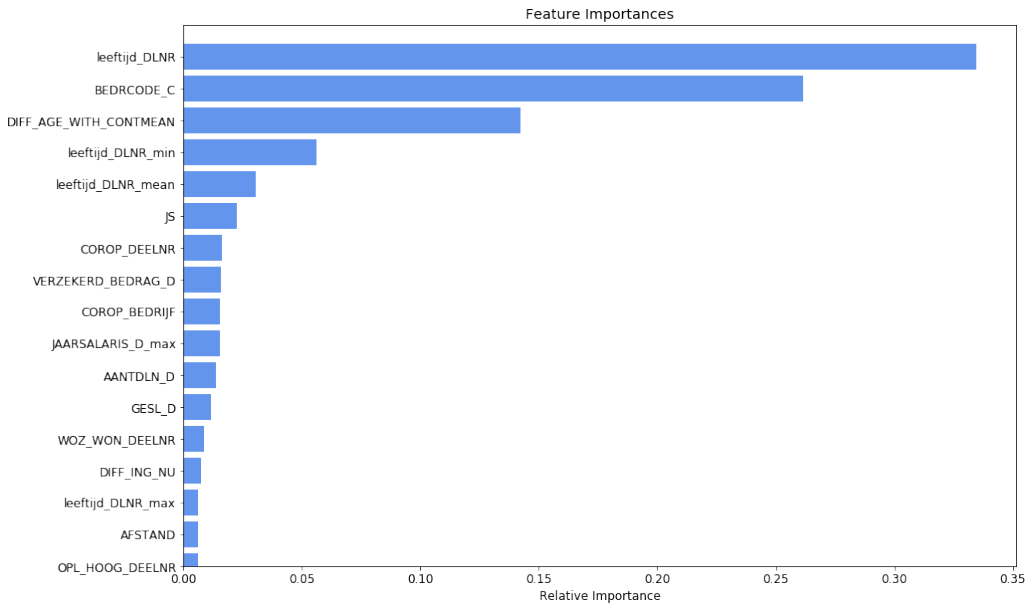

Figure 6.2: Feature importance of the GB model.

In Figure 6.2 the feature importance is displayed. The higher the relative importance, the more important the feature. Note that for readability the features with very low importance are not displayed. Although we do not display all features it should be noted that the GA algorithm uses 26 features in the total 1100 iterations it performs. The most important features correspond to the age and salary related features. This is in line with the current model. However, other features help improve the model. These include the amount of employees within a contract (”AANTDLN_D”), a company segment category (”BEDRCODE_C”) and geographical locations for the employee have a relatively large influence on the eventual

[26] Note, that we programmed this functionality ourselves as the package *sklearn* normalizes the feature importance per tree, thus giving a value of 1.0 per tree (representing one feature) as our tree depth is only one.

prediction of the model. Also interesting to note is the feature "GESL_D", which represent the gender of an employee. Self extracted features, such as distance between work and home ("AFSTAND") also has importance. The feature importance's indicate that most influence on the chance of a claim is based on employee data (age and salary). However, as shown by the feature importance company data also influences the predictions. In the old pricing model, company data had some influence. However, this was mainly used in a certain risk rate. These result show that it would be better to incorporate company features in the probability as well and apply geographical locations.

6.3 Partial dependence plots

Now that we have identified the most important features, we want to achieve an understanding of the impact these feature values have on the predicted probability. Within our high dimensional setting, which prevents us from plotting influence of multiple features at the same time, we use partial dependence plots. The partial dependence of a feature represent the average change in a prediction if changes occur in the particular feature while marginalizing out the other features [26]. This effect can be achieved through,

$$f(X_S) = \frac{1}{N}\sum_{n=1}^{N} f(X_S, x_{i,C}).$$

Here, X_S represents the partial dependence of feature S in which we change all values in feature S to a certain set value represented by X_C. The variable X_C takes the unique values present in X_S. We then compute the difference between the original predictions of the model and those of the model with feature values for feature S changes to X_C. For partial dependence plots we then average this computed change. In Figure 6.3 we visualize the partial dependence of the age feature. As a partial dependence plot visualizes the average curve of all points, individual interactions are not shown. In our complex data this might cause some records to overshadow interactions of the minority samples to changes in the feature X_S. To this end we also plot the individual conditional expectancy (ICE) which plots the functional relationships for individual observations [60]. To visualize the partial dependence and ICE plots clearer we center each value on the partial dependence of the first feature value of X_S.

We have plotted the partial dependence plots for the age ("leeftijd_DLNR") of an employee in Figure 6.3. We show a layout of a dashboard we have implemented. Within the dashboard a user can choose the feature(s) for which the partial dependence is plotted. Further one can use a feature for which the coloring of the ICE lines implies records with different binned values of this feature. Through this method we can inspect the dependencies between multiple features in the model. Furthermore, a user can specify a certain color scheme and a ratio of records to plot the ICE curves for.[27]

[27]The ratio is computed per binned value of the feature for which we color the lines. Besides retaining the distribution of values in the coloring feature this also prevents that some values of a feature might not be present at all in the ICE plot.

Within the ICE plots we color lines represented by employees that are female (1) by green lines and males (0) by blue lines. Furthermore, we zoom in on the average of the ICE curves, which corresponds to the partial dependence. We also show where most of the original feature values of the age in the data are present on the x-axis.

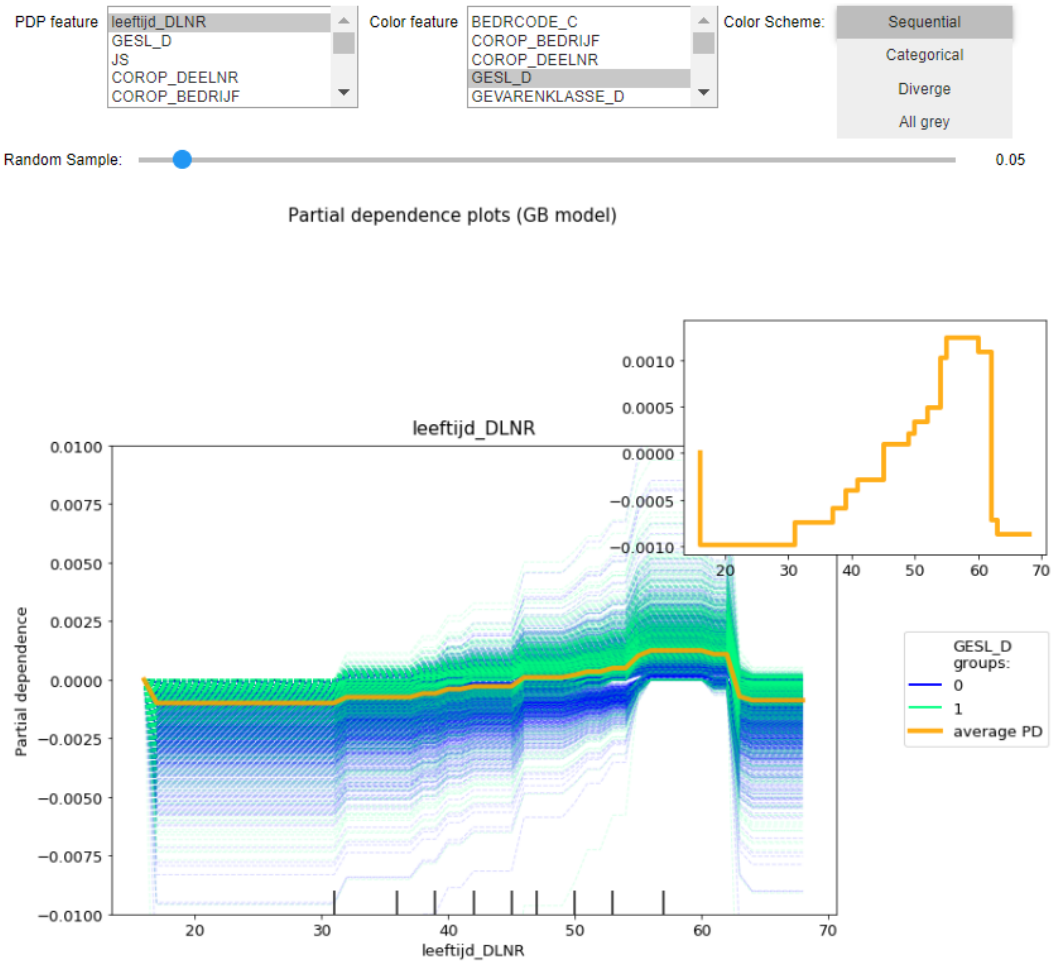

Figure 6.3: Partial dependence plots of the GB model.

By means of the ICE curves and partial dependence we can see how the predictions react to changes in the features. The results of age feature correspond to what is expected. The older an employee, the higher the probability of a claim. This relation is almost exponential instead of linear as is the case in the current model. If we look at coloring based on the gender feature, we see that the probability prediction for younger females is more insensitive and does not change much (green lines in the lower age bins are close to the center point of zero). Most males have a lower probability if the age is changed to low values. However, at around an age of 35 the average female probability is above the zero line and thus probabilities increase, while most male probability are still below the zero line. This is in line with what is expected (by the writer and others within Nationale-Nederlanden) as often females around these age bins have changes in their personal lives which can increase the probability of a disability. These results show how, unlike the current model, the GB model

can take dependencies between different features into account. The average partial dependence of age shows a drop in probabilities at an age of 63. Employees older than 63 thus get a lower probability resulting from the model. As the GB model (and other models) learn on training data and test on a test set, certain rules produced by the model might seem correct. However, this conclusion might be based on this particular training and test set and can lead to inaccurate probabilities for new groups of employees. The decision rule (in the decision trees within a GB model) at the age of 63 is produced by training on the training data and does lead to accurate probabilities for the test set. However we note that we have a finite dataset with a low number of claims. It can be the case that in this training and test set employees older than 63 have no claims, but new data may prove that this is not an accurate representation of the true probabilities. On the other hand, it can be the case that when employees are close to their pension age they often keep working. We note that the GB model sets the rules based on the training data and if the model is used in a pricing model one should assess whether these rules are in line with what one should expect. The partial dependence plots can assist in giving insights into the model rules.

In Figure 6.4 we also plot the partial dependence for the yearly salary (“JS”) and a geographical location (“COROP”) of the employee. The yearly salary partial dependence shows that the probability increases when salaries are below the maximum SV-loon threshold. However, we also see a decrease when we have very low salaries and very high salaries. We use the difference of the age with the contract mean in the partial dependence plot with a diverging color scheme (at zero difference it is white). This shows that, especially when a salary is below the SV-loon threshold and an employee is much older than the mean age of a contract, the probability increase is high. The partial dependence of the geographical locations clearly shows locations where the resulting probabilities are higher and where they are lower. These results can help with differentiating the probability, for instance on geographical locations, for a new pricing model. [28]

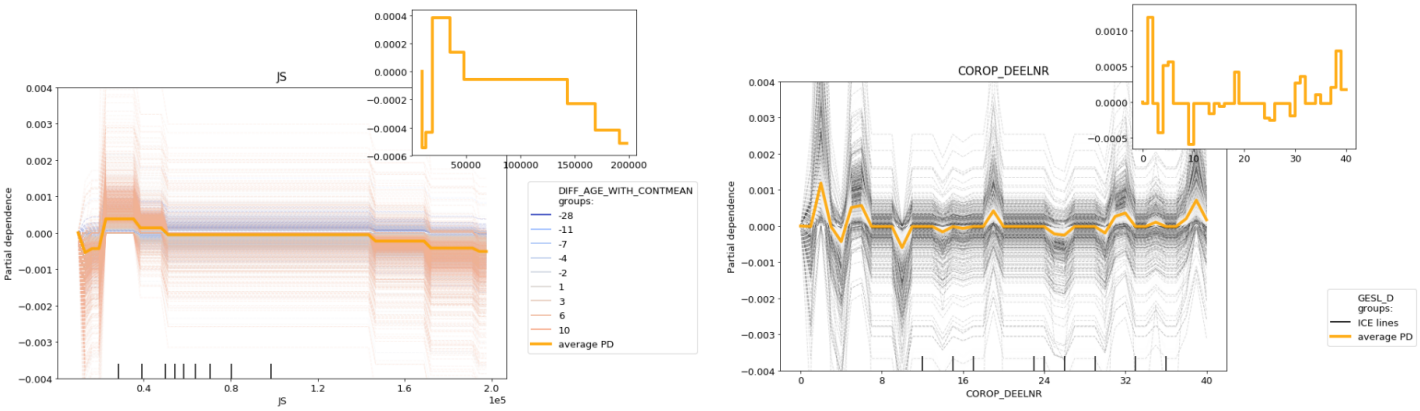

Figure 6.4: Partial dependence of the salary and geographical location of an employee.

[28] We will not discuss all the true results for privacy reasons in this report. We do show that changes in feature can influence the models prediction.

6.4 Improved probability accuracy

To assess the predicted probability accuracy with respect to the current model we implement the decrease in percentage of the BS between the current and new GB model. We look at the BS per binned feature values and check the percentage decrease (BS is better if it is lower) between the current (reference) and new model. We implement the gain we denote as, BS_{GAIN} through,

$$BS_{GAIN} = -\frac{BS_{model} - BS_{reference}}{BS_{reference}} * 100\%.$$

We check for this value over different values of a feature X_S, as we expect that for some binned employee groups the BS might differ. We plot the BS_{GAIN} in Figure 6.5.

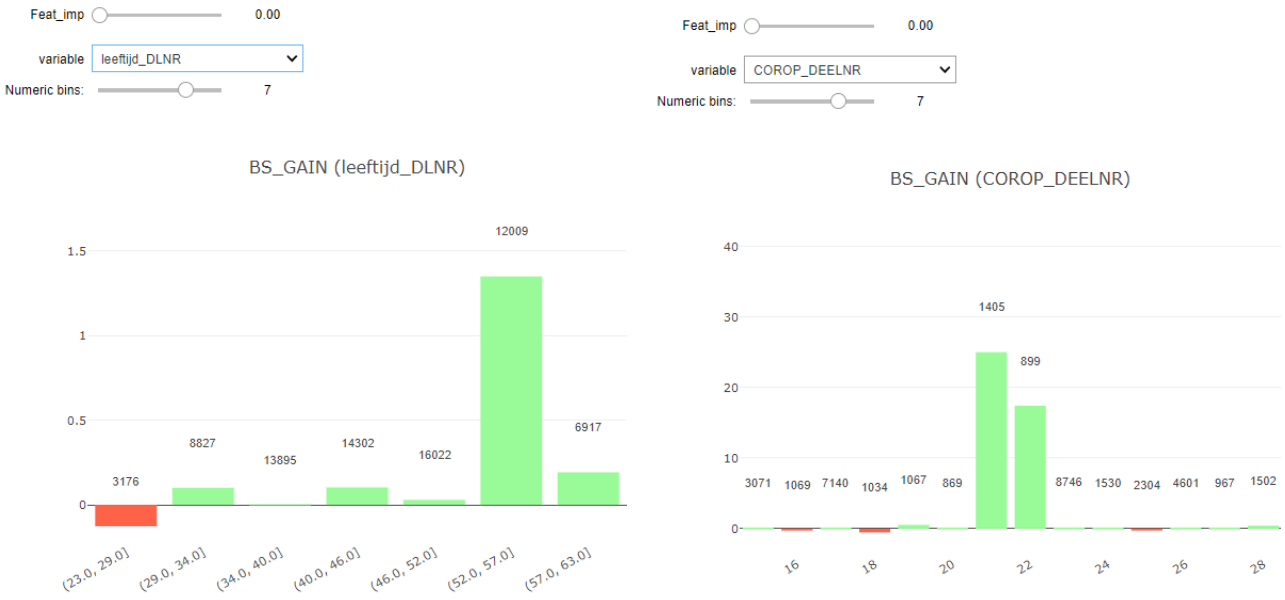

Figure 6.5: BS_{GAIN} of age and geographic location of employee (COROP).

We plot the BS_{GAIN} for the age of an employee ("leeftijd_DLNR") and a geographical location feature of an employee ("COROP_DEELNR"). We produced these images from a dashboard we produced. The user can alter the feature on display, the number of bins (for numeric features) and the minimal relative feature importance for the features in the drop-down menu. As can be seen from these plots, the BS, has improved in most bins due to the GB model. This can mainly be attributed to the higher probabilities given by the model to employees with a claim *and* lower probabilities for employees with no claim. By evaluating the gain in BS one can obtain insight into certain feature values in which inaccurate probabilities are present in the current model. For instance, for employees with a relatively high age, the old model gives substantially worse estimates than the GB model. This is due to the current model incorporating only age and salary. The probability estimates for the GB model in these higher age bins are differentiated on more features such as a geographical location.

Furthermore, we inspect the sum over the probabilities $P(y = 1|x)$ in the same bins for feature X_s. Here we apply the same dashboard structure and in addition give a hover text displaying the absolute sum of probabilities in each bin (visualized for the age feature in this plot). We display the sum over probabilities for the same features in Figure 6.6.

Figure 6.6: $P(y = 1|x)$ of age and geographic location of employee (COROP).

The summed probabilities of the age feature show that in the lower age bins the probabilities of the old model are a better fit. However, the number of records in these bins (displayed above the bars) is low. Overall the probabilities resulting from the GB model are more in line with the true observations. The summed probabilities of the geographic location feature show that adding this feature can help improve the resulting probabilities for employees in the different bins, especially bins with many records.

6.5 Implementation

In this section we outline the automated implementation of the machine learning process as described in previous chapters. The goal is to automate and simplify the entire process of data merging and pre-processing. Thereafter, the training process and prediction of the GB should be embedded in a simple sequential process. We aimed for an implementation which is both interpretable but also open to changes (in for instance the loss function).

In order to automate the merging process described in Appendix A, we use python package *cx_oracle* to initialize a connection to the data-warehouse of Nationale-Nederlanden. Another reason for this connection is the prevention of data leakage by downloading Comma-seperated-value (CSV) files and keeping these on a local computer. By asking the python user for its Oracle (SQL database) user-name and password a connection is made. These user-names are coupled to a Nationale-Nederlanden employees own account and thus re-

spects the privileges granted to that user within the data warehouse. Using a SQL query, the required tables are imported into the merging process. At the end of the merging process the new table is returned to the data warehouse using the connection, thereby eliminating the actual files on a local computer.

The merged data table contains columns which require human interaction. As some columns are not missing at random or need to be removed entirely. If these data preparation tasks are the same, as has been applied in this research, this is automated already. Thereafter, the MissForest algorithm (Section 4.1) can be directly applied to the remaining data. The null-values are removed automatically and the table can again be returned to the server. This Missforest algorithm can be used on any other dataset.

We have produced the GA algorithm in such a way that it works on any model. Hence, when using new models, or the GB model on new data (for instance other products) the GA algorithm can locate features with a high predictive power. We again kept an open structure to the algorithm. This way, the AUC of the PR-curve, on which we now calculate the fitness, can be swapped for any other metric.

If one wishes to retrain the GB model with different features, parameters or data, the training and validation process is parallelized for speed.[29] The model is trained over the different K-fold sets in parallel in order to decrease training time substantially. Furthermore, we opted to program the boosting algorithms (AdaB and GB) described in Chapter 5 from scratch. This allows for manually implementing original loss functions to gradient boost on. Furthermore, it allows us to create plots such as the ICE plots, which would not be possible if one would simply use a package. When a trained model has been obtained, we have written functions and produced dashboards in order to get the visualization and insights explained in this chapter. These dashboards include the feature importance, the BS_{GAIN}, the expected versus true plots as well as a dashboard to compare probabilities of different contracts and the rules applied in the model specific to one employee.

[29] Note, that at the time of writing, the structure for these evaluations is in place. However, a part of the python code still needs to be simplified and commented.

Chapter 7

Conclusions and recommendations

In this research we tried to improve the pricing model of the AOP product of Nationale-Nederlanden. To improve the pricing, the probability of an employee becoming disabled for work needed to be predicted with more accuracy than is the case in the current model. In Section 7.1 we report our findings and discuss these in Section 7.2. Concluding, in Section 7.3, we lay out recommendations based on the results obtained.

7.1 Conclusions

The goal of the research was to improve the pricing model of the AOP product. To this end we aimed to answer the main research question stating:

How can we utilize machine learning techniques to, more accurately, predict the inflow rate of individual AOP insured employees into the WIA and IVA regulation, thereby improving the pricing model of the AOP product of Nationale-Nederlanden?

We tested five distinct models to assess the performance on the imbalanced data of the AOP product. As the intent was to use the probabilities directly in a pricing model, the probabilities needed to be accurate in addition to a model being able to separate classes.

To quantify the increase in performance of the predictions, evaluation metrics where used. In current literature not many articles are concerned with the trade off between class separation and accuracy of probabilities. Although frequently used, the ROC-curve and the area under this curve, can give misleading results. We showed that in extreme imbalanced cases the false positive rate, which incorporates the true negative samples, was very insensitive to change due to the large number of true negatives. The precision-recall curve proved to be a better evaluation metric in these extremely imbalanced data settings were one is mostly interested in the minority records. With respect to evaluating the accuracy of probabilities we conclude that using one metric such as the Brier score in isolation gives an incomplete evaluation.

By assessing these evaluation metrics we conclude that all considered models had difficulties in separating the claim from non-claim data. The base models (Decision Tree, the Logistic Regression and Neural Network), without techniques to handle the class imbalanced,

performed poorly in terms of class separation and were slightly better than the current model. In literature, data re-sampling methods have been shown to increase metrics like the AUC of the ROC curve [33]. However, our research shows that although in some cases (especially if one applies SMOTE) this is true, these techniques have a negative impact on the probability estimates one obtains from machine learning algorithms. Training models on a different distribution caused the model to predict closer to the probability of the re-sampled minority class percentage if the model is not confident in its prediction.

We applied a method to transform the obtained probabilities by considering the ratio between the percentage of claims and non-claims in the training data and the test data. This method proved to be successful in some cases. However, one should be cautious by only observing the Brier score in these situations as transforming the probabilities obtained from a Decision Tree, which include probabilities of zero, led to a deviating sum over probabilities. Furthermore, the harsh assumption in this method that the distribution of the data does not change due to re-balancing the data caused inaccurate probabilities. Besides re-balancing we applied algorithmic solutions to the class imbalance in the form of boosting algorithms. Iterative adding decision tree models, designed to reduce errors in hard to classify records, proved to increase the class separation. The Gradient boosting model obtained the highest area under the precision-recall curve. The Adaboost model, although inferior to Gradient boosting, also obtained a high area under curve score relative to the base models.

The main difference in performance of the boosting models was the probability accuracy. The sum over the probabilities of the Adaboost algorithm diverged much from what is to be expected based on the test data. The same was true for the Neural Network. The probability calibration in the lower probability range is important for a pricing model in these extreme imbalanced situations. The Neural Network and Adaboost models produced confident probability predictions close to zero. This behavior is caused by the derivative of the loss function, in case of Adaboost, or the activation function, in case of the Neural Network. When training models in order to predict probabilities, one should be cautious of using functions which produce large gradients. Although class separability can be improved through this, it leads to probabilities close to zero and one when the increase in absolute model outcomes is transformed through the logistic sigmoid. The GB model performs well in terms of probability accuracy. The BS is low and and sum over the resulting probabilities is close to the actual occurrences. The histogram of probabilities showed a distribution similar to the Logistic Regression model. Hence, Gradient boosting on the logarithmic loss function proved to lead to adequate class separation while retaining accurate probabilities, leading to the conclusion to select this model for prediction purposes.

Besides the selection of a model the research focused on the selection of features with predictive power and the possibility of adding external data. We applied a Genetic Algorithm in order to search the feature space stochastically. The use of a wrapper feature selection model proved useful for most models. As the Genetic Algorithm is applied for each model separately the resulting number of features, and the features themselves, differ per model. Through a large decrease in the number of features, the separability between classes and the accuracy of probability predictions were increased for most models. During our research

we added a large number of external data features based on geographical locations of companies and employees. Through feature importance extraction based on the Gradient boosting model we can conclude that besides the age and salary of an employee, contract based and geographic location features can help improve the model. To help interpret the influence feature values have on the model results, we applied partial dependence plots with individual conditional expectancy lines. The partial dependence showed that, for the age and salary, the Gradient boosting model applies similar rules as the current model. Hence, while complying with the old model settings, the addition of other features did improve the performance.

Concluding, we showed that adding additional features and using more advanced techniques can improve the probability estimates of the AOP product. This leads to a more fair pricing per employee. Using far less features, while retaining some logical features, still produced adequate predicted probabilities and improvements upon the current model.

7.2 Discussion

Although the Gradient boosting model shows an improvement in performance over the current model, the precision is low for all recall values. This raises the question why this is the case and if improvements could have been made. In terms of data preparation the successful merging of the data proved to be cumbersome. Decision were made in order to keep the process automated. One of these decisions included the removal of records for which the claim date did not correspond with the date of the record. Although this did not lead to loss in claim data, it did remove some records which correspond to yearly data of employees which had a claim in later years. Although this is a very small part of the data, the information of employees before they had a claim might be very interesting. To automate the process we opted to remove these records, however this implies removing valuable information which might help improve the prediction.

Although time constraint was not an essential part of the decision between models and procedures it eventually proved to be problematic. This proved to be especially true when applying the Genetic algorithm to the boosting algorithms. Although the stochastic searching proved to able to decrease the features while improving performance for most models, the algorithm required a lot of time (days) in order to check the effects on the boosting algorithms. This time constraint of the Genetic algorithm for boosting models diminishes the usability for future projects.

Finally we note that machine learning models train on a subset of data. More advanced models, such as the Gradient boosting model, can find relations in features which lead to more accurate probabilities. However, these relations are not always very clear (which needs to be the case for a pricing model). For instance, during the research the performance was improved substantially but only due to a small sample of records with null-values accidentally being merged on records with a relatively high number of claims. Through feature importance's and the produced dashboards one can gain insight into the inner workings of the

model. Hence, we can find these faulty dependencies between the model outcome and the features but this requires investigation. In relation to this we emphasize that we have a finite dataset with a small number of claims. The Gradient boosting produced certain rules which might seem correct. However, this conclusion might be based on this finite dataset and can lead to inaccurate probabilities for new data. An example is the increase of probabilities with the increases with age of an employee. However, for records with an age higher than 63 the probability decreases. This rule is produced by training on the training data and does lead to accurate probabilities for the test set. One should be careful in adapting machine learning algorithms for a pricing model for which we have an extreme class imbalance as the inner workings of a model might apply to this particular dataset.

7.3 Recommendations

Training a machine learning model, often, requires a loss function to be minimized. Thought has to be placed in adapting the training process of machine learning models to account for the correctness of predicted probabilities instead of empirically determining this after training. Results showed that loss and activation functions which produce high gradients, push probabilities toward zero and one. We recommend that when one applies machine learning models with the intention of using the predicted probabilities directly, the binomial logarithmic loss function is used and activation functions which lead to large gradients need to be avoided. Furthermore, data re-balancing methods are to be avoided if the intent is to use the probabilities directly. Further research is needed to find better methods to transform probabilities after re-balancing data.

Although probability estimates are more accurate if more features are used, not all these features can be applied for producing pricing of products. By examining the exact thresholds set on features and the importance of those thresholds, decisions can be made on which features to include. In Section 7.2 we express the difficulties in finding the exact inner workings of a Gradient boosting model and the correctness of these rules based on data with so little claim records. We recommend using the produced dashboards in order to assess the inner workings on entirely new data as it becomes available (we do not have correct data past 2016). Furthermore, we advocate to use the new model for at least a year in the background in order to assess if the pricing does improve before adapting it as the new model. The produced Gradient boosting model can also serve as a way to see which groups of employees are currently being underpriced. If the Gradient boosting model is not applied for pricing purposes, feature relations and partial dependence in the model might help adapting the current model by adding new features or improve differentiation.

Finally, we strongly emphasize that data gathering should be done under some strict guidelines. If one key would be identical in multiple databases (here PICO and ASPRO), time to produce a dataset to train machine learning models for other products within Nationale-Nederlanden would be massively decreased.

Bibliography

[1] D. Autor, M. Duggan, and J. Gruber, "Moral Hazard and Claims Deterrence in Private Disability Insurance," National Bureau of Economic Research, Cambridge, MA, Tech. Rep., 6 2012. [Online]. Available: http://www.nber.org/papers/w18172.pdf

[2] W. Health Organization, "WORLD REPORT ON DISABILITY." [Online]. Available: https://www.unicef.org/protection/World_report_on_disability_eng.pdf

[3] UWV, "Maximumdagloon en maximummaandloon per 1 januari 2018 - UWV - Voor Particulieren," 2018. [Online]. Available: https://www.uwv.nl/particulieren/bedragen/detail/maximumdagloon

[4] Y. Yang, W. Qian, and H. Zou, "Insurance Premium Prediction via Gradient Tree-Boosted Tweedie Compound Poisson Models," 2016. [Online]. Available: https://arxiv.org/pdf/1508.06378.pdf

[5] S. Lee, K. Antonio, and S. C. Lee, "Why High Dimensional Modeling in Actuarial Science?" 2015. [Online]. Available: https://pdfs.semanticscholar.org/ad42/c5a42642e75d1a02b48c6eb84bab87874a1b.pdf

[6] J. Gray, "What Next? A Dozen Information-Technology Research Goals," 1999. [Online]. Available: https://arxiv.org/ftp/cs/papers/9911/9911005.pdf

[7] O. Maimon and L. Rokach, "Introduction to Knowledge Discovery and Data Mining," in *Data Mining and Knowledge Discovery Handbook*. Boston, MA: Springer US, 2009. [Online]. Available: http://link.springer.com/10.1007/978-0-387-09823-4_1

[8] Nationale-Nederlanden, "Brochure Collectieve Inkomensoplossingen Arbeidsongeschiktheid," 2014. [Online]. Available: https://adviseur.nn.nl/va/Zakelijk/Inkomensverzekeringen/WIA-Arbeidsongeschiktheidspensioen-1.htm#tab:undefined-3

[9] UWV, "Wat is sv-loon? - UWV - Voor Particulieren," 2018. [Online]. Available: https://www.uwv.nl/particulieren/veelgestelde-vragen/actueel/detail/wat-is-sv-loon

[10] ——, "Ik ben ziek (WIA-uitkering) UWV - Voor Particulieren," 2018. [Online]. Available: https://www.uwv.nl/particulieren/ziek/ziek-wia-uitkering/tijdens-wia-uitkering/detail/hoe-hoog-is-mijn-iva-uitkering

[11] Nationale-Nederlanden, "Product Approval Report WIA AOP tariff," Tech. Rep., 2014.

[12] Verbond van Verzekeraars, "Over het Verbond," 2018. [Online]. Available: https://www.verzekeraars.nl/het-verbond/over-het-verbond

[13] B. Djehiche and B. Löfdahl, "A hidden Markov approach to disability insurance," *eprint arXiv:1412.7334*, 12 2014. [Online]. Available: http://arxiv.org/abs/1412.7334

[14] G. A. Spedicato, C. Dutang, and L. Petrini, “Machine Learning Methods to Perform Pricing Optimization. A Comparison with Standard GLMs.” [Online]. Available: http://www.variancejournal.org/articlespress/articles/Machine-Spedicato.pdf

[15] V. Kašćelan, L. Kašćelan, and M. Novović Burić, “A nonparametric data mining approach for risk prediction in car insurance: a case study from the Montenegrin market,” *Economic Research-Ekonomska Istraživanja*, vol. 29, no. 1, pp. 545–558, 1 2016. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/1331677X.2016.1175729

[16] Y. Liu, B.-J. Wang, and S.-G. Lv, “Using Multi-class AdaBoost Tree for Prediction Frequency of Auto Insurance,” *Journal of Applied Finance & Banking*, vol. 4, no. 5, pp. 1–4, 2014. [Online]. Available: https://ideas.repec.org/a/spt/apfiba/v4y2014i5f4_5_4.html

[17] R. Roy and K. T. George, “Detecting insurance claims fraud using machine learning techniques,” in *2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT)*. IEEE, 4 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8074258/

[18] S. García, J. Luengo, and F. Herrera, “Tutorial on practical tips of the most influential data preprocessing algorithms in data mining,” *Knowledge-Based Systems*, vol. 98, pp. 1–29, 4 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705115004785?via%3Dihub

[19] J. L. Schafer and J. W. Graham, “Missing Data: Our View of the State of the Art,” 2002. [Online]. Available: https://stat.ethz.ch/education/semesters/ss2013/ams/paper/missing_data_1.pdf

[20] P. Gromski, Y. Xu, H. Kotze, E. Correa, D. Ellis, E. Armitage, M. Turner, and R. Goodacre, “Influence of Missing Values Substitutes on Multivariate Analysis of Metabolomics Data,” *Metabolites*, vol. 4, no. 2, pp. 433–452, 6 2014. [Online]. Available: http://www.mdpi.com/2218-1989/4/2/433

[21] H. Kang, “The prevention and handling of the missing data.” *Korean journal of anesthesiology*, vol. 64, no. 5, pp. 402–6, 5 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/23741561http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3668100

[22] D. J. Stekhoven and P. Buhlmann, “MissForest–non-parametric missing value imputation for mixed-type data,” *Bioinformatics*, vol. 28, no. 1, pp. 112–118, 1 2012. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btr597

[23] E. G. Armitage, J. Godzien, V. Alonso-Herranz, . López-Gonzálvez, and C. Barbas, “Missing value imputation strategies for metabolomics data,” *ELECTROPHORESIS*, vol. 36, no. 24, pp. 3050–3060, 12 2015. [Online]. Available: http://doi.wiley.com/10.1002/elps.201500352

[24] C. Penone, A. D. Davidson, K. T. Shoemaker, M. Di Marco, C. Rondinini, T. M. Brooks, B. E. Young, C. H. Graham, and G. C. Costa, “Imputation of missing data in life-history trait datasets: which approach performs the best?” *Methods in Ecology and Evolution*, vol. 5, no. 9, pp. 961–970, 9 2014. [Online]. Available: http://doi.wiley.com/10.1111/2041-210X.12232

[25] I. Brown and C. Mues, “An experimental comparison of classification algorithms for imbalanced credit scoring data sets,” *Expert Systems with Applications*, vol. 39, no. 3, pp. 3446–3453, 2 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741741101342X

[26] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning The Elements of Statistical Learning*, 12th ed. Springer, 2017. [Online]. Available: https://web.stanford.edu/~hastie/Papers/ESLII.pdf

[27] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006. [Online]. Available: https://books.google.nl/books/about/Pattern_Recognition_and_Machine_Learning.html?id=kTNoQgAACAAJ&redir_esc=y

[28] J. Howbert, "Classification: Basic Concepts, Decision Trees, and Model Evaluation," University of Washington Bothell, Tech. Rep., 2012. [Online]. Available: https://www-users.cs.umn.edu/~kumar001/dmbook/ch4.pdf

[29] V. Franc, A. Zien AlexanderZien, and B. Schölkopf, "Support Vector Machines as Probabilistic Models," Tech. Rep., 2011. [Online]. Available: http://www.icml-2011.org/papers/386_icmlpaper.pdf

[30] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 11 2016. [Online]. Available: http://link.springer.com/10.1007/s13748-016-0094-0

[31] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, p. 1, 6 2004. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1007730.1007733

[32] L. Yijing, G. Haixiang, L. Xiao, L. Yanan, and L. Jinling, "Adapted ensemble classification algorithm based on multiple classifier system and feature selection for classifying multi-class imbalanced data," *Knowledge-Based Systems*, vol. 94, pp. 88–104, 2 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705115004566

[33] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, 5 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417416307175

[34] J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera, "SMOTEIPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering," *Information Sciences*, vol. 291, pp. 184–203, 1 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025514008561?via%3Dihub

[35] S. Maldonado, R. Weber, F. F. I. Sciences, and u. 2014, "Feature selection for high-dimensional class-imbalanced data sets using Support Vector Machines," *Elsevier*. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025514007154

[36] G. M. Weiss, *Imbalanced Learning: Foundations, Algorithms, and Applications: Foundations of Imbalanced Learning*, 2012. [Online]. Available: https://pdfs.semanticscholar.org/1678/7e213ed0a5c0cf9baabdb45f9df631248a91.pdf

[37] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002. [Online]. Available: http://www.jair.org/media/953/live-953-2037-jair.pdf

[38] Y. Saeys, I. Inza, and P. Larranaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 10 2007. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btm344

[39] G. Menardi and N. Torelli, "Training and assessing classification rules with imbalanced data," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 92–122, 1 2014. [Online]. Available: http://link.springer.com/10.1007/s10618-012-0295-5

[40] S. Nagi and D. K. Bhattacharyya, "Classification of microarray cancer data using ensemble approach," *Network Modeling Analysis in Health Informatics and Bioinformatics*, vol. 2, no. 3, pp. 159–173, 9 2013. [Online]. Available: http://link.springer.com/10.1007/s13721-013-0034-x

[41] C. Zhang and Y. Ma, "Ensemble Machine Learning," pp. 11–16, 2012. [Online]. Available: https://doc.lagout.org/science/Artificial%20Intelligence/Machine%20learning/Ensemble%20Machine%20Learning_%20Methods%20and%20Applications%20%5BZhang%20%26%20Ma%202012-02-17%5D.pdf

[42] J. H. Friedman, "GREEDY FUNCTION APPROXIMATION: A GRADIENT BOOSTING MACHINE," *Statistics*, 1999.

[43] A. Niculescu-Mizil and R. Caruana, "Obtaining Calibrated Probabilities from Boosting," Tech. Rep. [Online]. Available: http://www.cs.cornell.edu/~caruana/niculescu.scldbst.crc.rev4.pdf

[44] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[45] T. Raeder, G. Forman, and N. V. Chawla, "Learning from Imbalanced Data: Evaluation Matters." Springer, Berlin, Heidelberg, 2012, pp. 315–331. [Online]. Available: http://link.springer.com/10.1007/978-3-642-23166-7_12

[46] I. H. I. H. Witten, E. Frank, M. A. M. A. Hall, and C. J. Pal, *Data mining : practical machine learning tools and techniques*. [Online]. Available: https://books.google.nl/books?hl=nl&lr=&id=1SylCgAAQBAJ&oi=fnd&pg=PP1&dq=missing+values+machine+learning&ots=8IBNqigBxb&sig=pAiUNLTScyxqSXy4nZqzVTcMY7g#v=onepage&q=missing%20values%20machine%20learning&f=false

[47] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: http://link.springer.com/10.1023/A:1010933404324

[48] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 12th ed., 2017. [Online]. Available: https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf

[49] J. H. Holland, C. Langton, S. W. Wilson, F. J. Varela, P. Bourgine, J. R. Koza, and A. B. Book, "Genetic Programming Complex Adaptive Systems Genetic Programming On the Programming of Computers by Means of Natural Selection," Tech. Rep., 1992. [Online]. Available: https://doc.lagout.org/science/Artificial%20Intelligence/Evolutionary%20computation/Genetic%20programming%20Complex%20adaptive%20systems%20-%20Koza%20J.R..pdf

[50] S. Cateni, V. Colla, and M. Vannucci, "Variable Selection through Genetic Algorithms for Classification Purposes," in *Artificial Intelligence and Applications*. Calgary,AB,Canada: ACTAPRESS, 2010. [Online]. Available: http://www.actapress.com/PaperInfo.aspx?paperId=37677

[51] H. Yin and K. Gai, "An Empirical Study on Preprocessing High-Dimensional Class-Imbalanced Data for Classification," in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*. IEEE, 8 2015, pp. 1314–1319. [Online]. Available: http://ieeexplore.ieee.org/document/7336349/

[52] H. Vafaie, K. D. J. T. w. A. Intelligence, . TAI, and u. 1992, "Genetic algorithms as a tool for feature selection in machine learning," *ieeexplore.ieee.org*. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/246402/

[53] R. L. Haupt and S. E. Haupt, “PRACTICAL GENETIC ALGORITHMS PRACTICAL GENETIC ALGORITHMS SECOND EDITION.” [Online]. Available: http://yag.es/Genetic-Algorithm/R.L. Haupt,%20S.E.Haupt%20-%20Practical%20Genetic%20Algorithms.pdf

[54] A. Irtaza, S. Adnan, K. Ahmed, A. Jaffar, A. Khan, A. Javed, and M. Mahmood, “An Ensemble Based Evolutionary Approach to the Class Imbalance Problem with Applications in CBIR,” *Applied Sciences*, vol. 8, no. 4, p. 495, 3 2018. [Online]. Available: http://www.mdpi.com/2076-3417/8/4/495

[55] Pencheva T, Atanassov K, and Shannon A, “Modelling of a Roulette Wheel Selection Operator in Genetic Algorithms Using Generalized Nets,” *BIOAUTOMATION*, vol. 13, no. 4, pp. 257–264, 2009. [Online]. Available: https://pdfs.semanticscholar.org/b349/5f6076557ca4fb20c0795fc373c4307602c6.pdf

[56] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 6 2002. [Online]. Available: https://jair.org/index.php/jair/article/view/10302

[57] M. Saerens, P. Latinne, and C. Decaestecker, “Adjusting the Outputs of a Classifier to New a Priori Probabilities: A Simple Procedure,” *Neural Computation*, 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.5247&rep=rep1&type=pdf

[58] R. Caruana, “Predicting Good Probabilities With Supervised Learning Alexandru Niculescu-Mizil,” 2005. [Online]. Available: https://www.cs.cornell.edu/~alexn/papers/calibration.icml05.crc.rev3.pdf

[59] skikit-learn, “sklearn.tree.DecisionTreeRegressor scikit-learn 0.19.2 documentation,” 2017. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

[60] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, “Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation,” Tech. Rep., 2014. [Online]. Available: https://arxiv.org/pdf/1309.6392.pdf

[61] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms David J.C. MacKay*, 2005.

[62] R. E. Schapire, “A brief introduction to boosting,” in *IJCAI International Joint Conference on Artificial Intelligence*, 1999.

[63] C. Li, “A Gentle Introduction to Gradient Boosting,” Tech. Rep. [Online]. Available: https://github.com/cheng-li/pyramid

Appendix A

Merging process

In this appendix we explain the steps taken in order to obtain a dataset which is used within the research. In Chapter 2 we explain how the dutch laws concerning disability is structured and describe the AOP product. Some of these insights are important to the merging process. We review all the steps taken in the merging process. On each step we visually show the steps taken in a flow diagram. In this diagram a number with a red circle corresponds to certain problems related to this step. These problems will be discussed in the text.

A.1 Merging data from individual databases

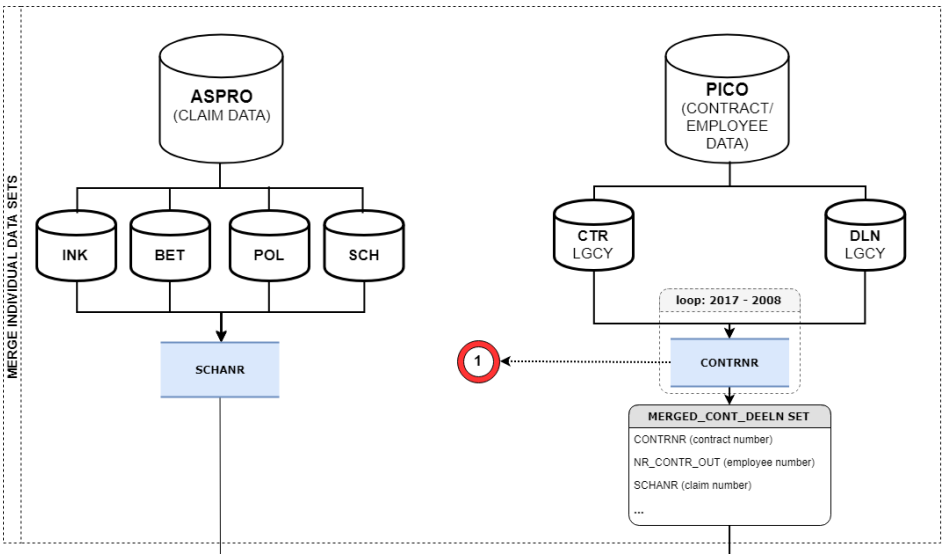

Figure A.1: First part of merging process.

In Figure A.1 we visualize the steps in the first part of the data merging process. The claim information is stored in the ASPRO data warehouse. We use information from the tables INK, BET, POL and SCH in this data warehouse. Within the INK table there is information on the employee who claimed. In the POL table the contract information for the particular claims is stored. The SCH table stores information on the claim itself such as the date of the injury. Finally, the BET table stored information on payments to the employee who claimed. Al these tables can be combined through the claim number, or “SCHANR”.

For the employee and contract data we extract information from the PICO data warehouse. We merge the two data tables within this warehouse on the contract number, or “CONTRNR”. The contract information for years 2006 till 2017 are stored in contract legacy table. The same hold for the employee data which is stored in the DLN (”deelnemer”) legacy table. We merge these tables per year (**1**) on the contract number, “CONTRNR”, which is available in both sets. At the end of this part we have a claim set with the correct information and a set on which for each employee, the correct contract data is coupled over multiple years (one employee might be in a contract for several years).

1. We merge each employee data to the contract information on a yearly basis. As later in the merging process we want to couple the claims to the correct employee and contract data. A claim occurs at a certain date and we want to have the data for this employee who claimed in the year that the claim occurred. We also want the contract information at the time the claim occurred (if we use later years the company might have moved location for instance) and as such we merge per year.

A.2 Merging data records with a claim number in the PICO data

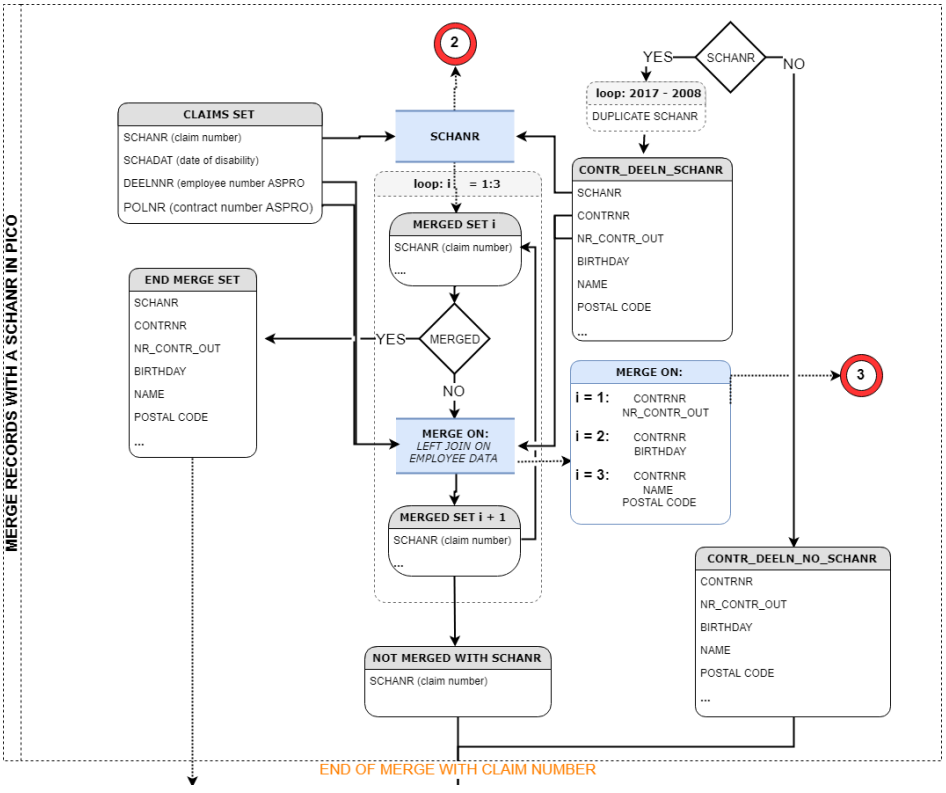

Figure A.2: Second part of merging process.

In the second part of the merging process we go follow the following steps:

The merged employee and contract data set is divided based on records containing a claim number and those that do not. The first merging step is to merge the claim set with the contract and employee set with a claim number (SCHANR) based on the SCHANR. The records within this set that are actually merged successfully are stored in the END MERGE SET. Merging on the SCHANR is the

most “save” way to merge the records. However, it is often not possible and we have to merge on different features. Furthermore, claim numbers are often added in the PICO data, years after the injury occurs (2).

2. In the employee and contract data obtained from the PICO warehouse, claim numbers are often not stored correctly or at all. These are added manually and for the AOP product, for which employees need to be injured for more then two years, claim numbers are often added many years later. As we want to merge the claim data, with a particular claim date, to the correct employee and contract data from which the claim occurred this leads to problems. As a SCHANR in the PICO data might not be present in the year a claim occurred. Hence, when a claim number is present for a particular employee we duplicate this claim number and add this in all the yearly data for the employee. Later in the merging process we delete falsely merged claims, for instance year of the data not corresponding to date of disability.

Records with a claim number, but which were not successfully merged, are merged in a loop. Within each loop we start with a set of records with a claim number which have not been merged. In each loop we merge on different features present in both data sets, which are ordered in a way that hopes to insure a correct merge (3). After each merge attempt, successfully merged (though they might contain errors) records are stored in the END MERGE SET. Records that are not, are again merged on a different feature set in the next loop iteration. We then obtain three sets. A set with merged records, a set of records with claim numbers but where not merged in this process and a set with records with no claim number in the employee and contract data. For the next merging part we concatenate the last two sets.

3. In the merging loop we merge on three different subsets of the features. We order these by a way that most likely lead to correctly merged records. After the claim number, the savest merge is on contract (“CONTNR”) and employee (“NR_CONTR_OUT”) numbers. The reason this “save” merge will not work for some instances is that employee numbers are manually added in the ASPRO data sets. The next merge is on contract number and the birthday (date) of the employees. It is a reasonable assumption that not many contracts have employees with the same exact birthday in the set. There after, we try a last merge on a contract number, employee name and its postal code. These last two merges will often not work as these fields are added manually into ASPRO as well. Hence, names might not be equal in both sets. At the end of the merging process we check for merged records with differences in birthdays and contract numbers from PICO and ASPRO data.

A.3 Merging data records with no claim number in the PICO data

In Figure A.3 the third part of the merging process is visualized. We check if remaining records, often with no claim number in the PICO data, can still be merged. These employees might had an injury but this was never registered in the PICO data. The merging process is similar to the merging of records with a claim number, although the programming is slightly different and hence, is a separate step in the merging process. Although not visualized here we concatenate the merged claims from second part of the merging process with records merged here. In the set with merged claims in the second part the non-merged records are also included.

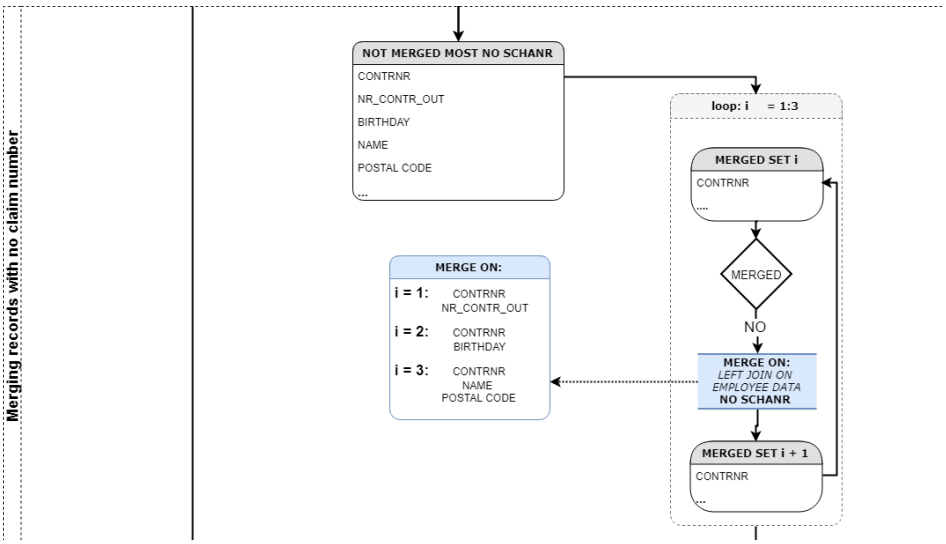

Figure A.3: Third part of merging process.

A.4 Cleaning the merged data set

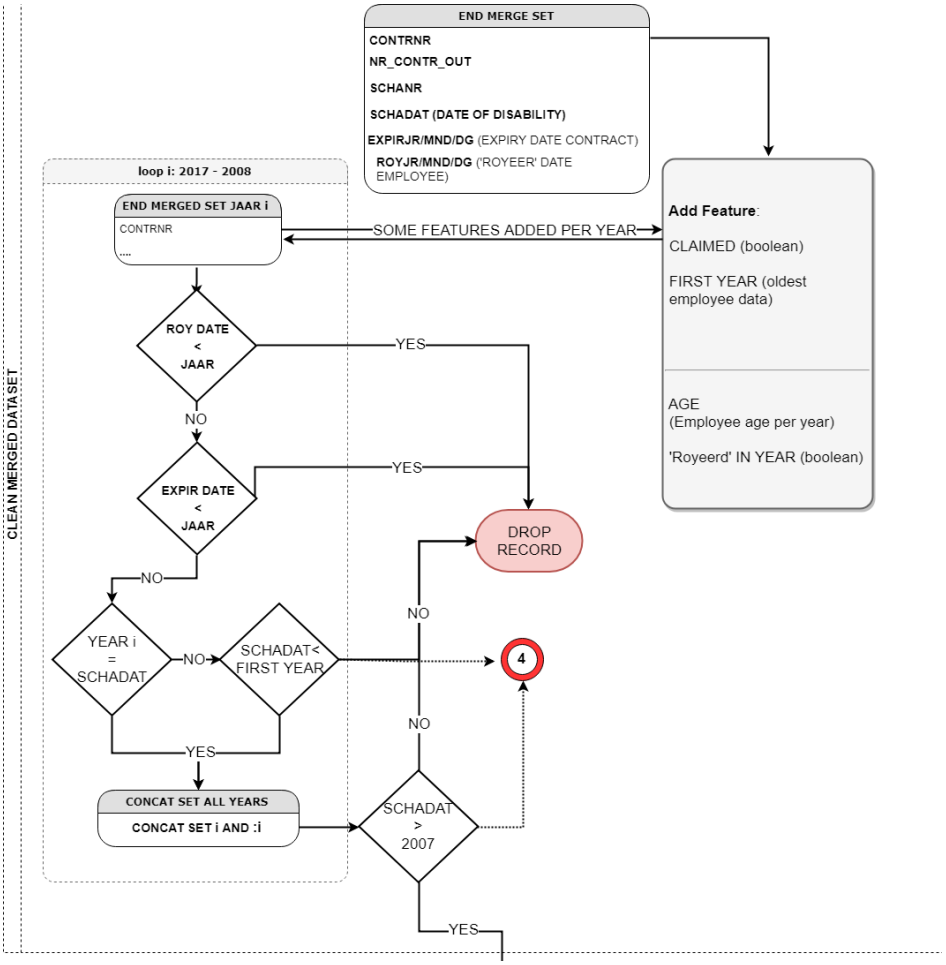

Figure A.4: Fourth part of merging process.

In the fourth part of the merging process we clean the obtained data set by removing claims that are, sometimes intentionally, merged incorrectly. We also add some additional features. We add a Boolean feature representing a claim or not has occurred. Furthermore, important for the merging process, we add a feature "FIRST YEAR". This represents the first year records from an employee are available. We start with the data set obtained in part three and use the expiration and "royeer" date from respectively the contract and employee data. If a contract is expired but is still in the dataset for a particular year we remove the records within this contract, the same hold for the "royeer" date. This is done in a loop over the years.

Thereafter, we remove all records with a claim for which the year the disability occurred does not correspond with the correct employee and contract data feature "JAAR", representing the years. This step is needed as we duplicated claim numbers and also do not merge based on years as this led to loss of correctly merged records (2). Within some records of a particular employee this led to discarding all the merged records. As the date of injury was before the first years of records available in the data. Hence, when the merged records of an employee has a disability date before the "FIRST YEAR" feature we discard all merged records expect the one corresponding to the first year of available data (4). Keeping records with disability dates before the first year of records does introduce the problem of disability dates before 2007. As we do not require data from these years we remove these records.

4. Records with a disability date before the year of the first data might be due to contracts being accepted and already disabled employees being signed as well. Furthermore, data might have been updated and previous yearly data might have not been recorded in the data warehouse.

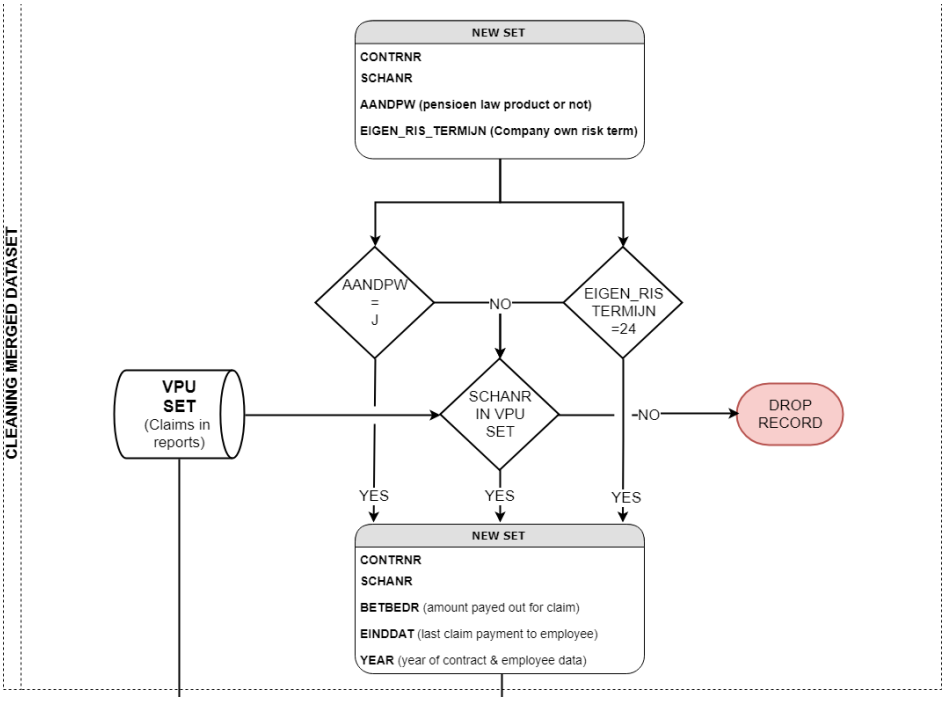

Figure A.5: Fifth part of merging process.

The next step of the merging process we remove all records, with *and* without claims, for which the features "AANDPW" and "EIGEN_RIS_TERMIJN" are not corresponding to the description of the AOP product. We did not remove these records before merging because these features can sometimes

be filled in incorrectly. By removing these records earlier some claims, corresponding to the AOP product (by looking at features in the ASPRO data warehouse) might be removed. Furthermore, we check if claim records have a claim number present in an external EXCEL file we denote as the VPU set. These are claim numbers corresponding to AOP claims as obtained from ASPRO. These records are known to be AOP claims. Hence, even when the "AANDPW" and "EIGEN_RIS_TERMIJN" feature are not correct we retain these claim records.

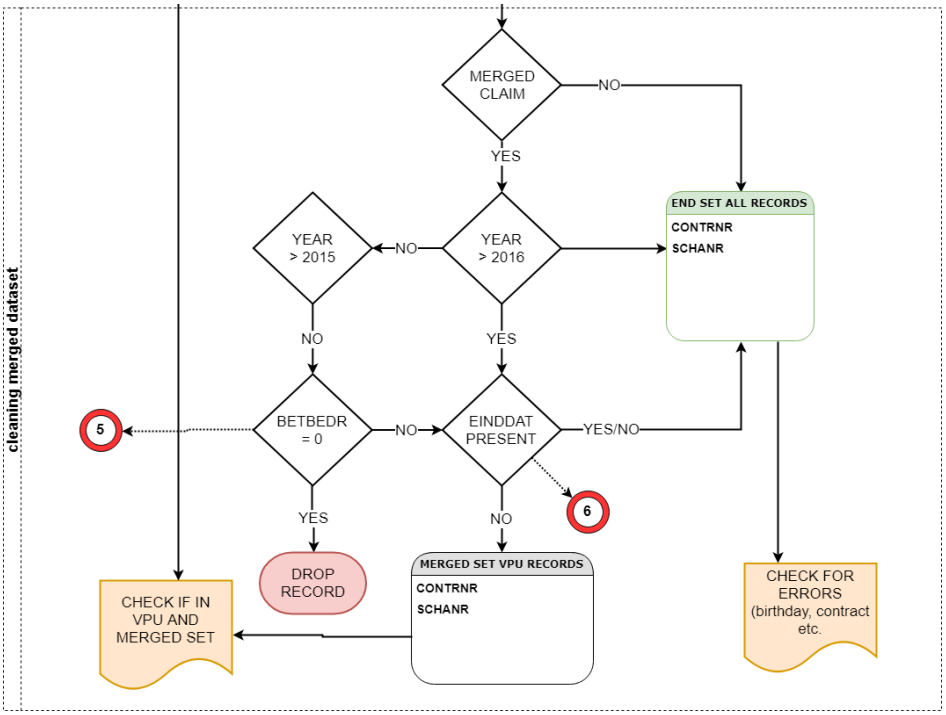

Figure A.6: Final part of merging process.

Starting with the set obtained from the last step we insert the non-merged records (records without a claim) into the final (non-preprocessed) dataset. For the merged records we look at some final conditions. We look at the features "BETBEDR" and "EINDDAT", revering to respectively the amount payed out to the employee who claimed and the last date of this payment. As for the AOP product to take effect employees need to be disabled for a least two years. A claim number or record might be in the ASPRO data which eventually did not end claiming. If no payment has been made to an employee with a claim date before 2016, this claim is removed (5). Furthermore, we look at the end date of these payments as to create a set which we can check with the VPU set (6). Eventually we insert all merged records which are left into END SET ALL RECORDS. We also incorporate the records with a present end date on the payments. Because we also include inactive claim records who at one point did receive payment. From the final set we export an Excel file in order to check for errors in the merged claims manually (the amount of claims is low). Possible errors include, non-matching birthdays or contract numbers from ASPRO and PICO.

5. An employee might get injured and recover within a few months. We remove these records from the set at this point as we look at inflow into the AOP product and not an disability perse. Furthermore, claim records with a disability date after 2015 are often not payed out yet as the two years have not passed and caused by lagging registration. As such we only look at "BETBEDR" for claim with a disability date before 2016.

6. In the VPU set we have records which are actual AOP claims obtained from the ASPRO data. We check if we have the same number of claims in the years 2008 till 2017 in our set in comparison with the VPU set. However, in the VPU set only active AOP claims are present. Hence, we check the end date of the payments to the insured. If no end date is present, then the records should still receive payment and should be present in the VPU set.

Appendix B

Data description

In this appendix the data is listed and a short description is given per feature. In Section B.1 we describe contract level data and in Section B.2

B.1 Contract level data

Table B.1: Contract level information.

feature name	**description**	**data format/ possible values**
CONTRNR	Primary key, contract number	integers
BEDRCODE	Contract industry sector	categories, encoded as integers
BEGCODE	Feature revering to payment clause in contract if claims are payed out to employee or company	K, to company — D, to employee
KORTCODU	Discount code, on amount of employees in contract	K, discount — O, ... — NaN
SEGMENTATIE_IC	Categories of point of sales track records in selling the product	string, nominal categories
WINSTDEL	Profit sharing between NN and insured company (large contracts)	N, no — J, yes — NaN
MAXUITDR	Maximum pay out period	integers, revering to days
STRAAT	Street where company is situated	string
WOONPL	Place name of the company	string
HUISNR	House number of the company	integers
POSTCODE	Postal code of the company	string
AANTDLN	Number of employees participating in the contract	integers
AANVPERC	Top-up percentage (percentage of salary repaid)	integers, revering to percentage
DUUR	Length of the contract	integers, revering to months
EXPIR-DG/MNT/JR	Expiry day/month/year of the contract	integers, revering to day/month/year
OORSPRING-DG/MNT/JR	Starting day/month/year of the contract	integers, revering to day/month/year
MAX_PW_SAL	Maximum pensionable salary within contract	float
MINVZBDR	Minimal insured amount	float
AANDPW	Product incorperated in pension law	boolean
GEINFVRL	Contract renewal automatic or not	boolean

MAILWPL	Mail address of company	string
TARGRSL	Premium tariff basis	category
AFWINC	Premium collection at VA or company	category
AANTAL_VOLGNRS	Number of employees in contract	integers
AFWPPROV	Deviant provision	integers
EIGEN_RIS_TERMIJN	Deductible term	integers, revering to months
ELFTDMAN	"AOW" age at signing of contract	integers, revering to years
MANTAD	Umbrella organisation or not (multiple companies)	boolean
MANTNAAM	Name of umbrella organisation	string
MIN_OBJECT	Minimal number of employees needed within contract	integers
NRAANTAL_GROEPEN	Number of groups within contract. Large companies with different conditions for groups	integers
POLMANT	Contract condition if umbrella organisation	category
SAL_BEGRIP	Conditions on how PW salary is build up	category
TPNR	Number of VA	integers

B.2 Employee level data

Table B.2: Employee level information.

feature name	**description**	**data format/ possible values**
NR_CONTR_OUT	Employee number within contract	integers
CONTRNR	Contract number	integers
GESL	Gender of employee	M, male — V, female
POSTCD_DLNM_PICO	Postal code employee	string
STRAAT_DNLM	Home street of employee	string
WOONPLAATS_DLNM	Home town of employee	string
AANDBTLND	Living abroad or not	boolean
GEVARENKLASSE	Risk category of employee	nominal category, encoded as integers
'SRTVERZ_D'	Category defining insurance type	category, encoded as integers
STATUS	Status of employee (precluded or not)	1, active — 3, precluded
ACHTERV	Suffix name of employee	string
EINDJR/MND/DG	"AOW" date of particular employee	integers, representing years\months\days
GEBJR/MND/DG	Birth date of employee	integers, representing years\months\days
GROEPNR	Number of group for employee within contract	integers
HUISNR_DLNM_PICO	House number employee	integers, nan
HUISNR_TOEV_DLNM	Suffix of house number employee	characters, nan
INGJR/MND/DG	Begin date of employee within contract	integers, representing years\months\days
JAAR	Year corresponding to data of this employee	integers
JAARSALARIS	Yearly salary of employee	float
MAX_VZBEDR	Maximum insured amount for the employee	float
VERZEKERD_BEDRAG	Insured amount of employee	float
BER_WYZE_VZBEDR	Method to calculate insured amount from yearly salary of employee	category
INDPOLISKENM	Contract charasteristic of employee	category

Appendix C

Mathematical derivations

C.1 Logistic sigmoid

Often we need to obtain the posterior probability $P(C_i|x), i \in 0, 1$. In multiple machine learning models this is obtained through working with the logistic sigmoid. Through Bayes rules we have that,

$$P(C_1|x) = \frac{P(x|C_1) * P(C_1)}{P(x|C_0) * P(C_0) + P(x|C_1) * P(C_1)}.$$

We can simply rewrite this as,

$$P(C_1|x) = \frac{1}{1 + \frac{P(x|C_0)*P(C_0)}{P(x|C_1)*P(C_1)}}.$$

If we define a as, [27]

$$ln\Big(\frac{P(x|C_1) * P(C_1)}{P(x|C_0) * P(C_0)}\Big),$$

We get,

$$P(C_1|x) = \frac{1}{1 + exp(-a)}. \tag{C.1}$$

Because $exp(ln(-a)) = \frac{1}{a}$. Equation C.1 is called the logistic sigmoid function and is often denoted by $\sigma(a)$. [27]

C.1.1 Derivative

Within multiple machine learning models the derivative of the logistic sigmoid function is using in order to apply gradient descent. The derivative of the logistic sigmoid function is,

$$\begin{aligned} \frac{\delta\sigma(a)}{\delta a} &= \frac{\delta}{\delta a}(1 + exp(-a))^{-1}. \\ &= -u^{-2}, \quad \text{with } u = (1 + exp(-a))^{-1} \end{aligned} \tag{C.2}$$

With,

$$\frac{\delta u}{\delta a} = -exp(-a). \tag{C.3}$$

We get,

$$\begin{aligned}\frac{\delta\sigma(a)}{\delta a} &= \frac{exp(-a)}{(1+exp(-a))^{-2}},\\ &= \frac{1}{(1+exp(-a))} * \frac{exp(-a)}{(1+exp(-a))},\\ &= \frac{1}{(1+exp(-a))} * \frac{1+exp(-a)-1}{(1+exp(-a))},\\ &= \frac{1}{(1+exp(-a))} * \Big(\frac{1}{1} - \frac{1}{1+exp(-a)}\Big),\\ &= \sigma(a)(1-\sigma(a)).\end{aligned} \tag{C.4}$$

Hence, the derivative of $\sigma(a)$ is simply $\sigma(a)(1-\sigma(a))$. This formula is easy and fast to compute.

C.2 Binomial cross-entropy loss function

As we are dealing with a binary classification (or binary probability estimation) problem, we often use the binomial cross-entropy loss function. This loss function is of the form,

$$L_{CE}(\hat{y}; y) = -\sum_{n=1}^{N} \Big(y_n * log(\hat{y}_n) + (1-y_n) * log(1-\hat{y}_n))\Big).$$

With $\hat{y}$ being the prediction of the model and y being the true target. The cross-entropy function originated in information theory. [61] It measure the uncertainty of a prediction. The cross-entropy loss function is a way of optimizing the negative log-likelihood. As we have a binomial target y we assume a binomial distribution. This is gives,

$$Pr(y_i; x_i) = x_i^{y_i} * (1-x_i)^{1-y_i}.$$

This is the joint distribution of y_i and x_i. As we assume independence between record, we can apply a likelihood function and get,

$$\begin{aligned}Pr(y|x) &= \prod_{n=1}^{N} Pr(y,x),\\ &= \prod_{n=1}^{N} Pr(y_i|x_i)^{y_i} * (1-Pr(y_i|x_i))^{1-y_i}.\end{aligned}$$

Taking the logarithm we get,

$$log(Pr(y|x)) = \sum_{n=1}^{N} y_i * log(Pr(y_i|x_i)) + (1-y_i) * log(1-Pr(y_i|x_i)).$$

Transforming this to a negative version is equal to the binomial cross entropy loss function. When this is zero, we obtain a maximum likelihood and thus, obtain the optimal solution.

C.2.1 Derivative

Multiple machine learning models require the derivative of a loss function. When using the Binomial cross-entropy loss function we have prediction $\hat{y_n}$ in the logistic sigmoid form of Equation C.1. We want to minimize this loss function by adapting prediction $\hat{y}$. The derivative is,

$$\begin{aligned}\frac{\delta E(\hat{y};y)}{\delta \hat{y}} &= \frac{\delta E(\hat{y};y)}{\delta \sigma(x_n)},\\ &= -\Big(\Big(\frac{y_n(\sigma(x_n)(1-\sigma(x_n)))}{\sigma(x_n)}\Big) + \Big(\frac{(-1)(1-y_n)(\sigma(x_n)(1-\sigma(x_n)))}{1-\sigma(x_n)}\Big)\Big).\end{aligned}$$

As the derivative of the logistic sigmoid function $\sigma(a)$ is $\sigma(a)(1-\sigma(a))$ (see Sub-section C.1.1) and by applying the chain rule with setting chain function as the logistic sigmoid. Writing this out gives,

$$\begin{aligned}\frac{\delta E(\hat{y};y)}{\delta \hat{y}} &= -\Big(y_n(1-\sigma(x_n)) - (1-y_n)\sigma(x_n)\Big),\\ &= (\sigma(x_n) - y_n),\\ &= (\hat{y_n} - y_n).\end{aligned}$$

C.3 Transforming probabilities after re-balancing

In Sub-section 5.3.2 we use the method proposed in [57] in order to transform the probabilities resulting from a model trained on re-balanced data. Here we derive the transformation formula used. We again mention that by applying Bayes theorem we have for our normal (test) data that,

$$P(x|y=1) = \frac{P(y=1|x)P(x)}{P(y=1)}. \tag{C.5}$$

The same applies for the resampled training data for which we denote probabilities by P_s. We want to have the transformed posterior probability $P(y=1|x)$. By applying Bayes theorem again we have that,

$$P(y=1|x) = \frac{P(x|y=1) * P(y=1)}{P(x)}. \tag{C.6}$$

At this point [57] makes the assumption that $P(x|y=j) = P_s(x_s|y=j), j \in \{0,1\}$. Meaning that the data distribution (distribution of x) does not change in the re-balanced set. We note that this is a rather harsh assumption if the undersampling rate s_{under} is not close to one. Furthermore, when using SMOTE we create synthetic data-points based on a few existing ones. We note that this also violates this assumption.

The next step is setting the term $f(x)$ as $\frac{P_s(x_s)}{P(x)}$. By using Equation C.6, the assumption and

Equation C.5 we get,

$$\begin{aligned} P(y=1|x) &= \frac{P(x|y=1)*P(y=1)}{P(x)}, \\ &= \frac{P_s(x|y=1)*P(y=1)}{P(x)}, \\ &= \frac{\frac{P_s(y=1|x_s)P_s(x_s)}{P_s(y=1)}*P(y=1)}{P(x)}, \qquad \text{(input Equation } C.5) \\ &= \frac{P(x)}{P_s(x_s)}*P_s(y=1|x)*\frac{P(y=1)}{P_s(y=1)}, \\ &= f(x)*\frac{P(y=1)}{P_s(y=1)}*P_s(y=1|x). \end{aligned} \tag{C.7}$$

As we approximately know $P(y=1)$ (it is a test set so we do not now the exact probability but we do know it based on the under- or oversampling rate s and the original distribution) we can use this formula to transform the probabilities. [57] uses the fact that, in our case $\sum_{j=1}^{2} P(y=j|x) = 1$, to get,

$$1 = \sum_{j=1}^{2} f(x)*\frac{P(y=j)}{P_s(y=j)}*P_s(y=j|x),$$

$$f(x) = \frac{1}{\sum_{j=1}^{2} f(x)*\frac{P(y=j)}{P_s(y=j)}*P_s(y=j|x)}. \tag{C.8}$$

From Equation C.8 and C.7 we get,

$$P(y=1|x) = \frac{\frac{P(y=1)}{P_s(y=1)}*P_s(y=1|x_s)}{\sum_{j=1}^{2}\frac{P(y=j)}{P_s(y=j)}*P_s(y=j|x_s)}. \tag{C.9}$$

Equation C.9 is the formula we use in order to transform the predicted probabilities resulting from training on a re-balanced data set.

Appendix D

Model parameters

In this appendix we list the model parameters and the results that led to these decisions. In Section D.1 we do this for the DT, in Section D.2 for the NN, in Section D.3 we review the AdaB parameters and in Section D.4 we examine the parameters of the GB model. We keep the LR model simple and do not adjust these parameters in order to show how a linear model behaves.

D.1 Decision tree

In a DT the main parameters to set are the impurity measure, the maximum tree depth and a possible minimal amount of records which should be present in the end leafs of a tree. To train the classification tree we use pythons *sklearn* package. Through literature research we found that the choice of impurity measure of a DT does not result in significant performance changes. [28] This due to the consistency between the measures. Consistency here means that the same choices are made in the splitting of nodes if the same class distribution is applied, see [28] for an explanation. As multiple other models considered in this research minimizes the binomial cross-entropy loss, we opted to use the entropy measure. The entropy measure if of the form,

$$Entropy_d = -\sum_{k=0}^{K-1} \hat{p}_{d,k} * log_2(\hat{p}_{d,k}).$$

Here d represents a decision node (see Sub-section 5.2.1). Note that for the binary class case this leads to,

$$Entropy_d = -(\hat{p}_{d,1} * log(\hat{p}_{d,1}) + (1 - \hat{p}_{d,0}) * log(1 - \hat{p}_{d,0})).$$

As $\hat{p}_{d,1} = 1 - \hat{p}_{d,0}$, due to the same amount of samples N_d in decision node d. Thus, this is similar to a binomial cross entropy measure. Note that for decision trees, the probability outcome is simply $\hat{p}_{leaf,1}$. Hence, as we do not want probabilities of 1.0 for claim cases, we set the minimal required amount of records in the leafs of the tree larger then the standard value of one. Furthermore, setting a larger value for this parameter lead to less overfitting in a tree. We set the minimum records in a leaf to 51 based on results on the validation sets. Setting a maximum tree depth also counters the overfitting on a training set. We saw that with high tree depths the performance on the validation set decreased. Through empirical results we set the maximum tree depth to seven.

D.2 Neural Network

NNs can capture non-linear relations between features and is, if correctly set up, very effective. The main disadvantage of NNs, as mentioned in Chapter 3, are the many possible different network structures. Besides the network structures the activation function taken can have a large impact. Due to time constraints we opted to use a randomized grid search over many possible network structures. For the NN we also use the *sklearn* package of python. Very small networks (2 layers with 2 hidden neurons) did not seem to be possible to capture the complexity of the data. A NN uses a gradient descent approach to adapt the weights through which forward propagation through the networks eventually gives a result. For multiple different network structures with different activation functions more hidden neurons gave better AUC results on the PR-curves. Through this process we eventually came to a network structure of (20,20,20). Meaning, twenty hidden neurons in each of the tree layers. This then leads to an output layer.

After we set this network structure a decision had to be made regarding possible activation functions. The activation functions have in impact on the gradient. This in turn, affects the updating of the weights. The activation function needed in the output layer was set in this research. As we are trying to predict a certain probability $P(y = 1|x) \in [0, 1]$ we use the logistic sigmoid function (see Appendix C). As is the case for the LR model, the output of the NN, in the form $w^T x$ (although adapted through previous activation functions) should be mapped to $[0, 1]$. The mapping applied for this is always the logistic sigmoid. Popular activation function include, besides the logistic sigmoid, the hyperbolic tangent, the rectified linear unit (ReLU) and a linear activation function. We did not consider the ReLU activation as it is often used for deep learning. The other activation functions are plotted in Figure D.1 for different values of a model outcome, here called x.

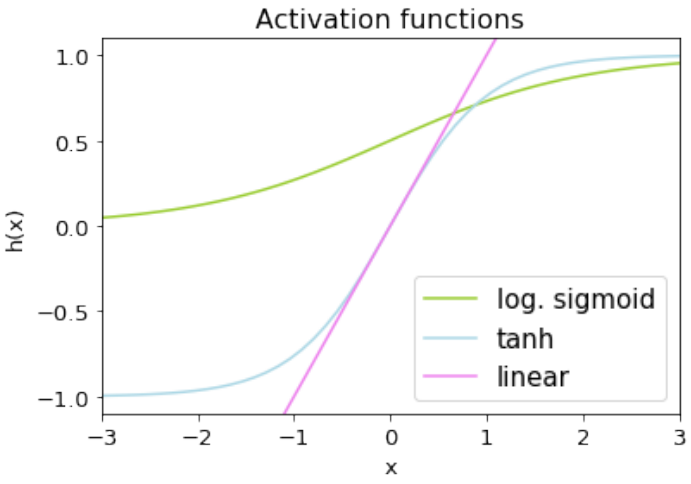

Figure D.1: Activation functions for the NN model.

Hence, the hyperbolic tangent activation function maps values within $[-1, 1]$. Through backpropagation, the weights w within each layer are adapter through the gradient of these activation functions. As can intuitively be seen in Figure D.1 (mostly from the logistic sigmoid function), negative output of the model in each layer will result in a predictions in favor of the negative class and the other way around. The intuition behind using the gradients to update weights is that very negative or positive outcomes of the models will have a gradient close to zero. As such weights will not be adapter much if predictions are very confident for one class. The gradient of the hyperbolic tangent

will be much larger, if predictions are less confident, resulting in larger weight updates. This allows the model to capture capture difficult records more accurately. Due to this behavior and corresponding (better) performance on the AUC of the PR-curve we opted to use the hyperbolic tangent function. Note that the gradient of the linear activation is always one or minis one. Using this function will not help use the NN structure to capture non-linear relations.

D.3 Adaboost

Although seemingly much more complex, the Adaboost algorithm as explained in Appendix E, does not have many adaptable input parameters. The learning rate a_τ is calculated through the model. Hence, the only parameter to tune is the amount of iterations T. In Sub-section 5.4.3 we already show the AUC and BS for the first 1500 iterations. In Figure D.2 we plot the results for 2000 iterations.

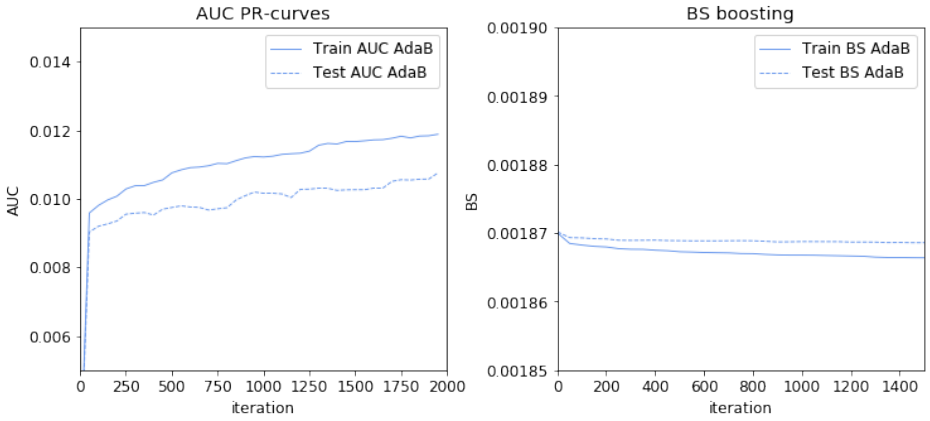

Figure D.2: AUC and BS for different values of T.

The AUC and BS still improve after 1500 iterations. As we still want the algorithm to set learning rate $a\tau$ itself (one can also set this equal in each iteration), we increase the amount of iterations to 2000 in order to increase the performance. Do note that this result in substantially increased training time for the model.

D.4 Gradient boosting

In Appendix F we describe the GB algorithm. The main parameters to set are the tree depth of the regression tree, the amount of iterations T and the learning parameter v. As boosting models are already complex, we opted to use tree depths of one (decision stumps). This way, if the model if fit, the rules of the tree can be obtained. Hence, the main decisions we regarding the amount of iterations to set and the learning rate v. We plot the increase in AUC and the BS for two values of v in Figure D.3.

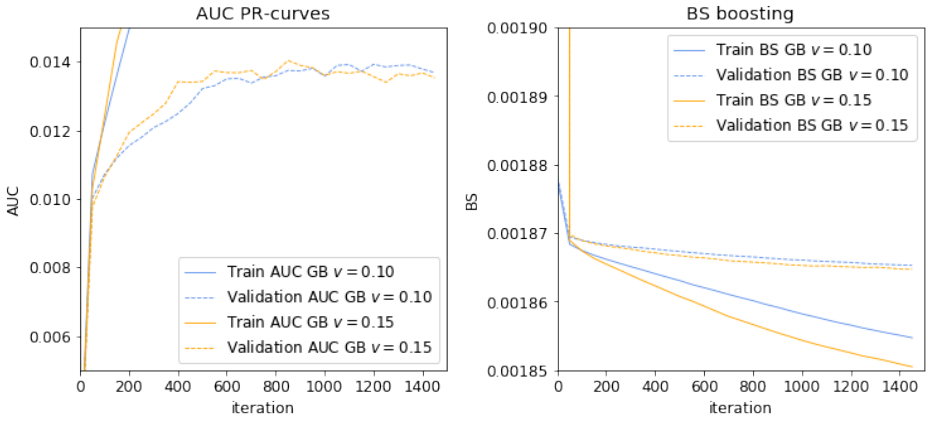

Figure D.3: AUC and BS for different values of v.

The learning rate v has a impact on the iterations needed to obtain a large AUC. This results in a faster increase in the AUC for larger values of v. With a larger value of v the BS also decreased faster. However, large values of v can cause the GB model to "shoot" past the global minima of the loss function and overfit on the training data. This is the case with $v = 0.15$ after approximately 900 iterations. Hence, using smaller values of v with more iterations is better. However, the time to train a GB model grows substantially by adding extra trees. The time to predict for test values does as well. We opted to set the parameter v low to 0.10 and the amount of iteration to 1100 as thereafter the increase is minimal on the test data and for some iterations we see a decrease.

Appendix E

AdaBoost algorithm

In this Appendix we outline the AdaBoost algorithm and the reasoning behind the different parameters and equations of the model. As mentioned in Section 5.4.1, AdaBoost is a linear combination of weak classifiers in the form,

$$G_T(x) = \sum_{\tau=1}^{T} \alpha_\tau g_\tau(x; w_\tau).$$

The pseudo-algorithm of AdaBoost, as originally implemented by [62], is outlined in Algorithm 3.

Algorithm 3 AdaBoost

INPUT: $X_{n,m}, n \in N, m \in M, y_n \in [-1, 1], n \in N$

procedure ADABOOST($X_{n,m}, y_n$)

Initialize weights $w_{n,1} = \frac{1}{N}$

for $\tau = 1$, $\tau{+}{+}$, while $\tau <= T$ **do**

Train weak learner $g_\tau(X, y, w_{n,\tau})$.

Obtain prediction $g_\tau(X) \rightsquigarrow [-1, 1]$, with error:

$$E_\tau = \sum_{n=1}^{N} w_{n,\tau} I(y_n \neq g_\tau(x_n))$$

Set

$$a_\tau = \frac{1}{2} log(\frac{1 - E_\tau}{E_\tau})$$

Update weight

$$w_{n,\tau+1} = \frac{w_{\tau,n} * exp(-a_\tau y_n g_\tau)}{Z_\tau}$$

▷ Z_τ is a normalization

Obtain $\hat{y}_n = sign(G_T(x_n))$

Obtain $P(C_1|x_n) = \sigma(G_T(x_n))$ ▷ σ refers to logistic sigmoid

The Adaboost algorithm improves the eventual prediction by focusing on the absolute error (from a binary prediction) obtained by the weak classifier g_τ multiplied by a certain weight. All information on which records x_n are hard to classify, based on previous iterations, is contained in the weights w_τ. To update the weights parameter a_τ is used. The main objective of the algorithm is to minimize the error on the training set at iteration, E_T, which is,

$$\begin{aligned} E_T &= \sum_{n=1}^{N} I(y_n \neq G_T(x_n)), \\ &= \sum_{n=1}^{N} I(y_n \neq \sum_{\tau=1}^{T} \alpha_\tau g_\tau(x_n; w_\tau)), \\ &= \sum_{n=1}^{N} I(\sum_{\tau=1}^{T} y_n \alpha_\tau g_\tau(x_n; w_\tau) < 0). \end{aligned} \tag{E.1}$$

This is true as we have that $y_n \in \{-1, 1\}$ and $g_T(x_n) \in \{-1, 1\}$. Using exponential rules we have that $I(z < 0) \leq Exp(-z)$ (if $z < 0, exp(-z) > 1$). Hence,

$$E_T \leq \sum_{n=1}^{N} \prod_{\tau=1}^{T} exp(-y_n \alpha_\tau g_\tau(x_n; w_\tau)), \tag{E.2}$$

Note that we thus can minimize the error in iteration T by minimizing the right part of Equation E.2.

The same reasoning of minimizing the error is applied in each iteration. In the first iteration the idea is to pick g_1 by minimizing E_1 with respect to a certain distribution $w_{n,1}$. The distribution represent the importance of each sample and is similar to giving a weighing an error on certain data records more heavily then others. We initialize $w_{n,1}$ as,

$$w_{n,1} = \frac{1}{N}.$$

Hence, we set each error to be of equal importance. The weighted error of the prediction in iteration one is,

$$\begin{aligned} E_1 &= \sum_{n=1}^{N} w_{n,1} I(y_n \neq g_1(x_n)) \\ &= \sum_{n=1}^{N} w_{n,1} I(\sum_{\tau=1}^{T} y_n \alpha_\tau g_\tau(x_n; w_\tau) < 0) \end{aligned} \tag{E.3}$$

However, we need to know how to update the weights to reduce the error in the second iteration. We know we can minimize the error by minimizing the right part of Equation E.2. Hence,

$$\begin{aligned} E_2 &\leq \sum_{n=1}^{N} exp(-\sum_{\tau=1}^{2} y_n \alpha_\tau g_\tau(x_n; w_\tau)) \\ &\leq \sum_{n=1}^{N} exp(-y_n \Big(a_2 g_2(x_n; w_2) + \sum_{\tau=1}^{2-1} \alpha_\tau g_\tau(x_n; w_\tau) \Big)) \\ &\leq \sum_{n=1}^{N} exp(-y_n a_2 g_2(x_n; w_2)) * exp(-y_n \sum_{\tau=1}^{2-1} \alpha_\tau g_\tau(x_n; w_\tau)) \end{aligned} \tag{E.4}$$

The last part does not change in iteration 2 and as such we set this as weight $w_{2,n}$. The same initiation holds if we look at E_T and hence we set,

$$w_{\tau,n} = exp(-y_n \sum_{\tau=1}^{T-1} \alpha_\tau g_\tau(x_n; w_\tau)).$$

Hence, to minimize the error in the next iteration we set the next weight $w_{\tau,n}$ to,

$$w_{\tau+1,n} = w_{\tau,n} * exp(-y_n a_\tau g_\tau(x_n; w_\tau)). \tag{E.5}$$

And we use Z_τ as a normalization over the weights in iterations τ. Meaning,

$$Z_\tau = \sum_{n=1}^{N} w_{n,\tau}. \tag{E.6}$$

We get, by taking sum out of the exponential,

$$Z_T = \sum_{n=1}^{N} \prod_{\tau=1}^{T} exp(-y_n \alpha_\tau g_\tau(x_n; w_{n,\tau})) \tag{E.7}$$

Which is equal to Equation E.2. Hence, the Adaboost algorithm is constructed in such a way that we have upper bound of the error E_T as,

$$E_T \leq Z_T.$$

Hence, to lower the error on the training data we should minimize Z_T. AdaBoost is a greedy algorithm that minimizes upper bound Z_T by picking a_T and g_τ optimally in each iteration as these are parameters we can change in Equation E.6. Hence, we do not optimize the error/loss on the training data as is normally the case with machine learning models. In Equation E.5 the exponential controls the weight updating. The weights are updated through,

$$exp(-y_n \alpha_\tau g_\tau(x_n; w_{n,\tau})) = \begin{cases} < 1, & \text{if } y_n = g_\tau(x_n; w_{n,\tau}). \\ > 1, & \text{if } y_n \neq g_\tau(x_n; w_{n,\tau}). \end{cases} \tag{E.8}$$

Hence, $w_{n,\tau}$ decreases if record n was correctly classified and increases if incorrectly classified with order a. Model, g_τ is optimized to train on target y with weights w_τ. The weight distribution is, as mentioned, normalized over n using Z. This gives us the updating formula in line 8 of Algorithm 3.

The one thing to still determine is parameter a_τ. We showed that AdaBoost seeks to minimize Z_τ in each iteration. From Equation E.6 and E.5 we get,

$$Z_\tau = \sum_{n=1}^{N} w_{n,\tau} * exp(-y_n \alpha_\tau g_\tau(x_n; w_{n,\tau})). \tag{E.9}$$

Taking the derivative with respect to a we get,

$$\begin{aligned} \frac{\delta Z_\tau}{\delta a_\tau} &= -\sum_{n=1}^{N} y_n w_{n,\tau} g_\tau(x_n; w_{n,\tau}) * exp(-y_n \alpha_\tau g_\tau(x_n; w_{n,\tau})), \\ &= -\sum_{y_n = g_\tau(x_n; w_{n,\tau})} w_{n,\tau} * exp(-a) + \sum_{y_n \neq g_\tau(x_n; w_{n,\tau})} w_{n,\tau} * exp(a). \end{aligned} \tag{E.10}$$

Equation E.10 is true as y_n and $g_\tau(x_n; w_{n,\tau}) \in \{-1, 1\}$. So when $y_n = g_\tau(x_n; w_{n,\tau})$ we get a value of one, and if $y_n \neq g_\tau(x_n; w_{n,\tau})$ a value of minus one is obtained. With keeping in mind that the weights represent the eventual error we get,

$$\frac{\delta Z_\tau}{\delta a_\tau} = -(1 - E_\tau) * exp(-a) + E_\tau * exp(a). \tag{E.11}$$

As we want to minimize Z_τ we set the derivative to be zero. This results in,

$$\begin{aligned} (1 - E_\tau) * exp(-a) &= E_\tau * exp(a) \\ (1 - E_\tau) * \frac{1}{exp(a)} &= E_\tau * exp(a) \\ \frac{(1 - E_\tau)}{E_\tau} &= exp(2 * a) \\ \frac{1}{2} * ln(\frac{(1 - E_\tau)}{E_\tau}) &= a. \end{aligned} \tag{E.12}$$

This is equal to the formula used in Algorithm 3 in line 7. Do note that if the error E is equal to 0,5, which corresponds to a random prediction, a becomes zero and as such the resulting tree prediction is not useless.

E.1 Adaboost's exponential loss function

Several years after the introduction of the Adaboost algorithm [42] proposed a statistical framework for boosting in general and showed that Adaboost minimizes the exponential loss function. The exponential loss function with $y \in \{-1, 1\}$ is of the form,

$$L_{Exp}(G(x); y) = exp(-yG(x)).$$

We showed that Adaboost minimizes Z_T which is given in Equation E.13. By again putting the product over τ back into the exponential we get,

$$Z_T = \sum_{n=1}^{N} exp(-y_n \prod_{\tau=1}^{T} \alpha_\tau g_\tau(x_n; w_{n,\tau})). \tag{E.13}$$

By looking at the exponential loss function form we see that by decreasing Z_T (Equation E.9)in each iteration, we are actually decreasing the exponential loss function which for one record n represent,

$$\begin{aligned} L_{Exp}(G(x); y) &= exp(-y \sum_{\tau=1}^{T} \alpha_\tau g_\tau(x; w_\tau)), \\ &= exp(-yG_T(x)) \end{aligned} \tag{E.14}$$

We already showed that taking the derivative of Z_τ and setting this to zero results in the parameter a_τ. Hence, parameter a_τ represent the gradient direction and step taken in order to minimize the exponential loss.

Appendix F

Gradient boosting algorithm

In this appendix we outline the Gradient boosting algorithm and the reasoning behind the different parameters and equations of the model. The Gradient boosting algorithm is of the form.

$$G_T(x) = \sum_{\tau=1}^{T} v_\tau g_\tau(x; \gamma_\tau).$$

Note that in our case we set v_τ equal for each iteration. The pseudo-algorithm of Gradient boosting is outlined in Algorithm 4.

Algorithm 4 Gradient boosting

INPUT: $X_{n,m}, n \in N, m \in M, n \in N, y_n, v \in [0, 1]$

procedure GRADIENT BOOSTING($X_{n,m}, y_n$)
  Initialize $G_0(x_n) = log(\frac{y \in 1}{y \in 0, 1-y \in 1})$ ▷ We base the starting point in relation amount $y \in 1$.
  for $\tau = 1$, $\tau{+}{+}$, while $\tau <= T$ **do**
    Calculate $\gamma_\tau = -\frac{\delta L(y, G_{\tau-1})}{\delta G_{\tau-1}}$
    Train regression tree $g_\tau(X)$ on γ_τ
    Predict $\hat{y}$
    $G_\tau = G_{\tau-1} + v * \hat{y}$
  Obtain $\hat{y}_n = sign(G_T(x_n))$
  Obtain $P(C_1|x_n) = \sigma(G_T(x_n))$ ▷ σ refers to logistic sigmoid

As mentioned in Section 5.4 we want to improve eventual model G_T by adding predictions of a certain model in order to get closer to the target y. Or,

$$G(x) + g(x) = y.$$

As we need to find model g in each iteration we rewrite this as (note that $G_{\tau-1}$ was the previous prediction,

$$g_\tau(x) = y - G_{\tau-1}(x). \tag{F.1}$$

The term $y - G_{\tau-1}(x)$ is called the residual [63]. We want to have g_τ set so that it can reduce this residual (hence, learning on hard to classify points). Note that we use regression trees as weak

learner models g_τ. In our gradient boosting machine we are training on the binomial logarithmic loss function. Thus our loss function is,

$$L(y, \hat{y}) = -\Big(y * log(\hat{y}) + (1 - y) * log(1 - \hat{y}))\Big)$$

.

As we aim to minimize this function we take the gradient with respect to the parameter we can change, namely $\hat{y}$. Note that $\hat{y}$ is the outcome of model $G_{\tau-1}(x)$. In Appendix C we show that the negative gradient of this loss function corresponds to,

$$(G_{\tau-1}(x) - y_n) \tag{F.2}$$

Note that this is the negative of Equation F.1. Hence, to find optimal weak learner g_τ we train on the negative gradient of the logarithmic loss function.