PREDICTING RISK: TRUMPING STANDARD PRACTICE IN AUTO INSURANCE

Master's Thesis
MSc Econometrics and Management Science
Business Analytics & Quantitative Marketing

Rotterdam, December 20, 2018

Written by

Wendelien Bakelaar
369051

Supervised by

Dr. A. Alfons (ESE)
Dr. A.E. van Heerwaarden AAG (EY)
Drs. S. Tolk AAG (EY)

Erasmus School of Economics
Erasmus University Rotterdam

I would like to express my gratitude to Dr. Angela E. van Heerwaarden AAG and Drs. Sander Tolk AAG for their enthusiasm and for sharing their actuarial expertise which helped provide relevant context to this research. I would also like to thank Dr. Andreas Alfons for his faith in my abilities throughout this process. Lastly, I would like to thank my mother for supporting me, even when she did not really understand what I was doing.

Abstract

This thesis presents an approach for identifying extremes in auto insurance by means of risk profiles. A standard generalised linear model serves as benchmark whereas gradient boosting and backpropagation neural networks are considered as alternatives. To handle class skew, the latter methods are combined with Multi-Minority SMOTE, an alteration of standard SMOTE presented in this thesis, as well as cost-sensitivity. Considering the Area under the ROC curve, the ability to identify the high risk class and the depth of insights provided, gradient boosting combined with MM SMOTE is deemed to be the most appropriate method for predicting risk.

Contents

List of Figures

List of Tables

1 Introduction

Individuals purchase insurance because they want to insure themselves against risk, but by removing the risk from themselves they are in fact imposing this risk on their insurer. Given the number of customers one insurance company may have, this accumulated risk soon becomes immense. For this reason, it is essential that insurance entities are able to identify the factors which are most likely to contribute to an individual's propensity to cause damages and to, to some extent, develop risk profiles. This research aims to do just that.

The general insurance market is both highly competitive as well as strictly regulated, which prevents insurance companies from setting excessive prices and furthermore, ensures that insurances are available to everyone. In many countries, especially in Western Europe, it is even enforced by law that every individual must be insured and in some contexts, companies are unable to refuse anyone who fills out the application. Despite this rigidity, however, a new development is taking place. Due to a combination of increased data storage as well as improved computation power, more and more advanced data-driven techniques are being applied in this sector. As a result, the market is becoming accessible for non-traditional companies with Google being the prime example[1].

The reason these data-driven techniques are deemed successful is their ability to distinguish between the risk levels of customers. Customers have their own perceptions about their level of recklessness, and the less these perceptions are reflected in the premiums they pay, the more likely they are to switch to another insurance company. By making use of new, data-driven techniques which allow us to capture complex and non-linear relations, we are better able to identify the customers who are most likely to incur the highest costs. In the ideal setting, premiums could even be tailored specifically to an individual based on his/her risk classification, but even without this ability, having a better understanding of what drives the likelihood of a claim can help insurance entities better predict the sum of liabilities and associated risk margin, also known as technical provisions. Furthermore, these insights allow for increased transparency in the insurance market as compared to having a simple standard premium with no information as to how it was determined. The process of determining accurate risk profiles is therefore relevant for both the insurance entities as well as their customers. On the one side costs are minimised and on the other side customers receive premiums which better reflect their risk profile.

Research Question: *To what extent can data mining techniques identify extremely risky auto insurance policy holders? In addition, to what extent can these methods provide insights regarding the nature of these policy holders?*

This thesis investigates the potential for new data-mining techniques, more specifically boosting and neural networks, in identifying high risk profiles in car insurance. The aim of the investigation is to determine whether new data-mining techniques can trump standard practice, involving Generalised Linear Models (GLM), in identifying the individuals who are most likely to cause substantial damages. The individuals are divided into three risk profiles; low risk (no claims), medium risk (infrequent or mediocre claims) and high risk (frequent and extreme claims), where the latter is of particular interest. Given that the distribution of classes is imbalanced (claims and particularly extreme claims are not common), this research

[1] Google wants to collaborate with French insurance entities in order to provide data-driven insurance premiums (Reuters, 2016). By cooperating with existing insurance entities, Google can circumvent the high entry costs to the market and can soon pose a threat to existing companies.

also has academic relevance by providing an empirical application of techniques designed to deal with imbalanced data. In addition, whereas most of these methods have been tested on standard and mostly binary datasets, this application involves a more substantial and multi-class dataset. The imbalance methods considered are the newly introduced Multi-Minority Synthetic Minority Oversampling Technique (MM SMOTE) as well as cost-sensitivity. These methods are applied in combination with multinomial logistic regression, which serves as a benchmark, a backpropagation neural network and gradient boosting.

The data used for this research is the automobile insurance portfolio of a Dutch insurance entity spanning a period of two years. The data contains information about the policy holder, the policy itself, the vehicle owned, the claims made (if any), as well as an overview of demographic factors based on the individual's zip code, for a group of over one million policy holders. In addition, for the 128,009 claims made, the time of the claim as well as the corresponding claim amount are known. Lastly, the portfolio considered is that of private passenger insurance and hence excludes commercial or public transport vehicles. Given how substantial and high-grade the data is, this dataset lends itself perfectly to investigate the potential for data mining in auto insurance. However, there was one important limitation related to the data, namely that of security. The data could only be used in combination with a secure (and unstable) server, narrowing down the computational capacities. Many issues could be circumvented by adjusting existing functions and breaking up the code into smaller chunks, but as a result of the security measures a lot of time went into restructuring code and some concessions had to be made, as will be discussed in the text.

2 Literature Review

The insurance market has always had to deal with a variety of issues, many of which are related specifically to this industry. In the Netherlands, for example, the law states that every individual who owns a car must have at least the minimum amount of insurance. This insurance, referred to as WA-insurance (or third party liability insurance), guarantees that you will be covered for any damages you cause to others, or their vehicle, whilst driving. The issue with having such a law is that premiums must be determined objectively, however, these premiums must simultaneously be fair for the majority of the population, which covers a broad range of risk profiles. Insurance entities would prefer to only serve the low risk segment of the population but in order to do so, they must first be able to identify the high risk individuals and secondly, be able to deter them. The latter is often achieved by setting undesirably high premiums, as this is easier than having to explicitly reject a customer, whereas the first part; identifying those most likely to incur high costs, will be discussed in this thesis. Once consumers have joined a particular insurance company, they can then choose to purchase a diverse selection of additional insurances, and will do so according to their risk aversion, disposable income and self-perceived level of risk.

On the other hand, standard economic theory suggests that perhaps it is risk-seeking behaviour which drives the purchase of additional insurance. This is a result of asymmetric information in which one party knows more than the other. In this case, the essential information is the level of risk taken by the driver and if he knows himself to be reckless, he will be more inclined to purchase additional insurance. A more specific form of asymmetric information is moral hazard, a term which has become almost synonymous with insurance since Arrow (1963) first introduced it. Moral hazard in motor insurance implies that individuals are more prone to careless behaviour once they are insured. This is because the consequences

of an accident are not as severe, at least financially, if you are insured. A variety of research has been done to test the presence of these problems in the (motor) insurance market, yet the results have been quite mixed. Whilst Chiappori and Salanie (2000) find that there is no relation between accident frequency or severity and contract chosen, Cummins and Tennyson (1996), Abbring et al. (2008) and Cohen (2005) find evidence proving the opposite.

These concepts are important because they are related to risk. The more risky a customer is, the more likely he is to have an accident and hence drive up the costs for his insurer. As an indicator of risk, insurers often consider two measures, namely, the frequency and severity of car insurance claims (Desyllas and Sako, 2013). The frequency is defined as the number of claims divided by the exposure (i.e. the number of days of insurance coverage considered), whereas severity is the total value of claims divided by the number of claims incurred in a particular time period. The product of these two measures is an indication of the risk premium required for an insurance agency. To illustrate:

$$Budget = \frac{number\ of\ claims}{exposure} * \frac{total\ claims\ amount}{number\ of\ claims} = frequency * severity. \quad (1)$$

The product of these measures can be expressed simply as the total claim value divided by the exposure, however, frequency and severity are found to behave quite differently and hence they are generally investigated separately (Brockman and Wright, 1992).

In order to predict either the frequency or severity of claims, standard practice dictates that the Generalised Linear Model (GLM, explained in more depth in Section 4.1) should be used (Anderson et al., 2004). Given that claims may be quite rare, especially in a limited time frame, frequency is often limited to a binary variable indicating whether or not there was a claim. Moreover, the Poisson distribution is often found to be most appropriate for this measure, whereas the Gamma distribution is most suitable for severity. David (2015), Mosley Jr (2004), Ohlsson and Johansson (2010) and De Jong et al. (2008) have all demonstrated the merits of GLMs in the context of auto insurance.

Whilst there is an abundance of literature proving the effectiveness of GLM in private passenger insurance, very little has been done to investigate the effect of data-mining techniques in this field. Given the recent developments, including a market shift towards data-oriented companies, it is essential to have a better understanding of the potential these methods have in the insurance market. Chapados et al. (2002), for example, compares the accuracy of various machine learning techniques to that of GLM in determining car insurance premia. They find that Support Vector Machines (SVM) yield disastrous results as this method focuses on the conditional median whereas the conditional mean is preferred. On the other hand, he finds that neural networks, and in particular a mixture of neural networks, produce significantly better results than standard methods. This can be attributed to this method's ability to differentiate the risk premia amongst clients. Guelman (2012) makes use of gradient boosting, a method he praises for being highly accurate, requiring little preprocessing and, in contrast to other machine learning techniques, has interpretable results. He applies gradient boosting in the context of insurance loss cost modelling and finds that this method has significantly better predictive power than GLM does.

In addition to data mining, there is also the matter of imbalanced data in auto insurance. The vast majority of policy holders do not have any claims during a particular period,

meaning that the majority of the data is assigned a low-risk profile. This implies that techniques which predict that every policy holder is relatively risk-free will be exceptionally accurate, yet will also be irrelevant. For this reason, the methods used in this research field must account for the inherent imbalance in the data. As imbalance is particularly detrimental for data mining techniques, and the implementation of these techniques in auto insurance is quite limited, it naturally follows that previous literature combining imbalance and auto insurance is scarce. However, imbalanced data can be found in many and diverse applications, some of which being fraud detection, disease diagnostics and the identification of potential churners, i.e. leaving customers.

In these areas the techniques for dealing with imbalanced data can be divided into three main categories. The first is over- and/or undersampling, the second is cluster-based sampling and the third is an algorithmic adjustment, typically involving cost-sensitivity. Undersampling in its simplest form is done at random, in order to minimise the ratio of majority to minority classes. This method has been used in insurance by Guelman (2012), in intrusion detection by Qazi and Raza (2012) and on a variety of UCI datasets by Tahir et al. (2009). Undersampling proved to be quite effective in these contexts, however, in the case of severe imbalance, a substantial amount of data and hence information must be removed. Another approach is to make identical copies of minorities, i.e. oversampling, but this soon leads to overfitting, as proven by Chawla et al. (2002). In response to this finding, these authors developed their own method called the Synthetic Minority Oversampling Technique (SMOTE), which combines undersampling and synthetic oversampling (for more details see Section 4.4.2). Many have developed their own adaptations of SMOTE, however, the original method remains most popular (Chawla et al., 2003; Han et al., 2005a; Lusa et al., 2013; Wang et al., 2006). Cluster-based undersampling methods, on the other hand, have the advantage that they do not randomly discard observations, but rather make a selection of observations from each of the majority class clusters. This method works nicely in reasonably sized datasets, however, soon becomes infeasible in the context of big data (Yen and Lee, 2009; Rahman and Davis, 2013).

The final approach, namely cost-sensitivity, incorporates the fact that some errors are more severe than others. Failing to identify a high risk individual is more costly than falsely accusing someone of being high risk. This is because the premiums an insurance company sets reflect the risk, i.e. claim size and frequency, they expect from this customer. Failing to identify a precarious individual implies that the premiums they receive from this individual will likely not cover the expenses this individual makes. Vice versa, if they overestimate someone's risk they will charge more premium than necessary for this client. The cost involved here is that this person may purchase their insurance elsewhere. Whether or not cost-sensitivity is applicable in a particular setting depends on whether the costs of various errors are measurable. If this is the case, then cost-sensitivity is often applied by adjusting the loss function of a particular method. This can be done in virtually any method, but is especially common in machine learning. Although this method often provides excellent results, there is little consensus on which approach is best, especially since many methods still struggle with parameter tuning, multi-class data or other aspects of the implementation. Nevertheless, Kukar et al. (1998), Beijbom et al. (2014), Sun et al. (2007), Zheng (2010), Fan et al. (1999) and Zhou and Liu (2006) all provide interesting implementations in this field of research.

Building on work discussed in this section, this thesis aims to investigate the possibilities for data mining techniques in private passenger insurance, specifically in identifying and

characterising high risk policy holders. Given the success of boosting and neural networks in capturing complex relations, these two machine learning methods are considered in this research. In addition, in order to account for the class imbalance, these methods are also applied in combination with MM SMOTE, an adjustment of SMOTE introduced in this thesis, and with adjusted cost-sensitive loss functions. To assess the merits of these methods, their results are compared to those of GLM, more specifically multinomial logistic regression, which serves as a benchmark. The following section describes the data used in this thesis. Section 4 explains and discusses the data mining techniques as well as the methods for dealing with imbalanced data are discussed. This thesis will then continue to present the results in Section 5 and ends with the conclusions and suggestions for further research in Section 6.

3 Data

The data considered in this thesis have been provided by a Dutch insurance entity. The data, spanning from 2014 to 2015, include the records of over one million policy holders from one specific car insurance policy. By considering only one insurance policy, a degree of homogeneity is present between the individuals[2]. The data which is collected directly by the insurance company is quite limited; only the individual's date of birth, gender, zip code and license plate number are recorded at the time of purchase. However, this data has been extended by means of two external datasets, one based on demographic factors, the other based on characteristics of the car. The zip-code based variables are province, urbanisation, social class, income level, education level, home type and home ownership. These variables are not provided on an individual basis but instead indicate the averages in a particular street, i.e. zip code. The features related to the individual's car are its brand, weight, power, fuel type, gearbox, catalogue value, distance driven and years owned. Finally, the coverage type as well as the number of damage-free years (which is updated each year) of the policy holder are known. Since the data used are confidential, the use of summary statistics is avoided and the variables will henceforth be anonymised, referring only to, for example, *Individual Variable A*, *Car Variable B* or *Social Variable C*.

Given that the aim of this research is to identify the most risky clients, a measure for risk has been determined. If one would attempt to estimate the extremes based simply on claim size, then there is a substantial risk for overfitting; the largest claims can lead to the largest errors and hence in a regression scenario it would likely be optimal to include dummy variables for these particular observations or otherwise hinge on particular attributes which are unique to these observations. This, however, would not reveal general patterns and hence the results could not be extrapolated to new clientèle. For this reason, this thesis makes use of risk profiles. The various policy holders are divided into one of three risk profiles; low-risk (no claims), medium-risk (one claim of mediocre size) and high-risk (multiple claims or one exceedingly large claim). The threshold for an extreme claim is set to the third quartile of all claim sizes considered.

Before assigning the policy holders to classes, however, several preprocessing steps have been taken, the most essential being aggregation. As mentioned in Section 2, standard practice dictates that GLM be used in the field of insurance, and in addition, GLM in combination with an offset. An offset is a variable which is used to weight observations in a

[2]This homogeneity stems from the fact that selection bias, i.e. inherent differences between groups based on their choice of insurance, can be ignored when only considering one type of policy.

GLM, and in this case is often set to the exposure level. The reason for doing so is that car insurance data is often divided into small pieces, as can be seen in Table 1. At the beginning of a policy holder's contract (which can be any day of the year), and every time a policy holder purchases a car, has an accident or changes his situation in some way, shape or form, a new row is added to the data. The exposure measures the duration of each observation, hence by incorporating it in the form of an offset, you are making an observation of one year comparable to an observation of one month. However, due to anomalies in the data collection, the splitting of observations is not as pure as it should be. An example is shown in Table 1 where a new row is made just for the accident and moreover, the associated exposure is set to 0.0001 whereas the exposure for one day should be 0.0027. Due to these types of errors (and more) in the exposure, the data are aggregated, rather than using an offset. The data are aggregated based on the individual, car and year, implying that there is a unique observation for each combination of policy holder, license plate and policy year, hence removing the need to incorporate exposure.

Table 1: Example data

StartDate	EndDate	PolicyID	ZipCode	DateOfBirth	CarID	Cat.Value	Weight	DFYears	Claim	Exposure	ClaimDate
01Jan2010	07Apr2010	111111111	9999ZZ	01Jan1960	12345	11395,00	820	10		0,26557	
08Apr2010	31Dec2010	111111111	9999ZZ	01Jan1960	21212	9390,00	839	10		0,73374	
01Jan2011	31Dec2011	111111111	9999ZZ	01Jan1960	21212	9390,00	839	11		0,99932	
01Jan2012	31Dec2012	111111111	9999ZZ	01Jan1960	21212	9390,00	839	12		1,00205	
01Jan2013	31Dec2013	111111111	9999ZZ	01Jan1960	21212	9390,00	839	13		0,99932	
01Jan2013	31Dec2013	111111111	9999ZZ	01Jan1960	21212	9390,00	839	13	1000,00	0,0001	01Jan2016

Furthermore, in order to ensure that the performance measures are measured adequately, even though the size of the data already somewhat guarantees this, the data is split into three different combinations of training and test sets by means of k-fold cross-validation, where k equals three. In k-fold cross-validation, the data is randomly divided into k different "folds", i.e. subsets, and each fold is at some point used as the test set. The remaining folds function as training set which leads to k different sets to be used at input. This concept for k equal to three is illustrated in Figure 1. It is common to consider five or even ten folds in cross-validation, however, due to the sensitivity of the server connection as well as the substantial computing time, three was deemed more appropriate. The required model parameters are determined by running another level of three-fold cross validation within each of the tree training sets and evaluating the performance therein[3]. Once these parameters were determined, the models are run on each of the three training sets in their entirety and then predictions are made on their corresponding test sets. The final performance measures are averaged over the three. Moreover, to make sure that the categorical variables are balanced over the three sets, infrequent values are merged with others. This prevents the scenario that all observations with a particular value for, e.g. car brand, are all in the same fold.

[3]Consequently, there is no separate validation and test set, but instead the hyper-parameters are determined based on their performance within each set of training folds and then evaluated on each corresponding test fold. Having a separate validation set may have lead to a purer evaluation of the techniques.

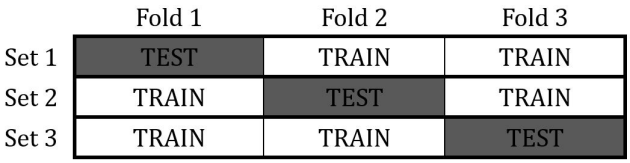


Figure 1: Illustration of 3-fold cross-validation.

Once the data is aggregated and the folds are made, the individuals are assigned to one of the three risk profiles. Those with no claim in a particular year were assigned the low risk profile. Then, as agreed with experts in the field, individuals with either three or more claims or an annual claim amount which exceeded the third quartile were assigned the high risk profile. It follows that all remaining observations were deemed medium risk. Given that the dataset is rather massive, the proportions of the classes in each fold is the same to three decimal places. Hence, Table 2 shows the proportions of the three classes across all three folds.

Table 2: Proportion of classes across all three folds.

Low Risk	92.5%
Medium Risk	5.6%
High Risk	1.9%

3.1 Missing Data Imputation

As is usual in large datasets, the data considered contains missing values. This is not such a problem when dealing with categorical variables as the missing values can easily be assigned their own class label, e.g. *Unknown*. This is done for the variables *Car Brand*, *Gender*, *Social Class*, *Fuel Type*, *Gearbox* and *Car Body Type*. How to deal with missing numerical values, on the other hand, is a highly debated topic. The simplest solution is to get rid of all incomplete observations, however, this is equivalent to discarding information. Not only is this not the most sophisticated approach, it may also lead to biased estimates if the missing observations are missing not at random (Kang, 2013). Another solution is to fill in the missing values with either the mean or median of this particular variable. The disadvantage of this approach is that it is a simple one dimensional strategy which ignores all other information contained in the observation. Consequently, the missing values are imputed by means of k-Nearest Neighbours (kNN).

This method, explained in more depth in Section 4.4.2, starts off by identifying the most similar observations, i.e. neighbours, for each incomplete observation. The missing values of this observation are then replaced by the median value of this variable in its closest neighbours. The median is chosen as this is a more robust measure than the mean, which is highly sensitive to outliers. KNN imputation therefore makes use of the non-missing information in a particular observation in order to impute its missing values. In order to implement kNN, the *VIM* package in R is used, however, this method is not directly implementable on a dataset which is as substantial as the one considered in this

thesis[4](Kowarik and Templ, 2016).

In order to circumvent this problem and because the missing numerical variables are all car characteristics (*Power*, *Vehicle Age*, *Catalogue Value* and *Weight*), kNN is applied per car brand. This implies that the neighbours in the imputation method belong to the same car brand as the observation under consideration. Furthermore, given that the provided information for a particular person is not directly related to the characteristics of his/her car, only the car specific variables are included in the imputation. Hence, missing values for a BMW car are imputed based on the most similar BMW cars in the data. The reason for choosing kNN is that this method can be used to impute both numerical and categorical variables and does not require specifying a model per variable being imputed (Edgar Acuna and Rodriguez, 2004). Moreover, it suits this relatively simple purpose of linking similar cars to one another for extracting information.

4 Methodology

4.1 Generalised Linear Model

Generalised Linear Models (GLM) are a generalisation of the standard linear model, which are characterised by the conditional distribution of the response variable. Various models fall under the GLM umbrella and hence GLM is applicable in remarkably diverse contexts. In the case of multinomial data, the appropriate GLM is multinomial logistic regression.

4.1.1 Multinomial Logistic Regression

Multinomial logistic regression is a regression technique where the dependent variable is nominal and hence it is a suitable model for tackling multi-class classification problems. As with any model, there are various assumptions which should be met, although in the case of multinomial logistic regression, these are quite limited; only the assumption of independence of class membership (the membership of one class is not related to the membership of another class) and the assumption of non-perfect separation (the outcomes cannot be perfectly separated by the regressors) are required (Starkweather and Moske, 2011). The procedure for this method is based on binary logistic regression, which can be extended to a multi-logit model. While there are many implementations for this model, this thesis considers the penalised maximum likelihood approach as it incorporates regularisation parameters.

In GLMs, the desired probabilities are typically determined by means of linear predictor functions $f(k, i)$ (Friedman et al., 2010). These functions, as shown in Equation 2, are used to determine the probability that a particular observation i belongs to class k,

$$f(k, i) = \beta_{0,k} + \beta_{1,k} x_{1,i} + \beta_{2,k} x_{2,i} \tag{2}$$

where $\beta_{j,k}$ is the coefficient associated with regressor j and class k, and $x_{j,i}$ is the value of regressor j for observation i. Merging these coefficients into coefficient vectors per class β_k, leads to the following $K - 1$ logits:

$$\log \frac{\Pr(G = k|x)}{\Pr(G = K|x)} = \beta_{0,k} + x^T \beta_k, \quad k = 1, ..., K - 1. \tag{3}$$

[4]R cannot allocate enough memory to make a square matrix with close to 2.8 million rows on the laptop provided for this research.

Using a more symmetric approach, as put forth by Zhu and Hastie (2004), the class membership probabilities can then be modelled as follows:

$$\Pr(G = k|x) = \frac{\exp^{\beta_{0,k}+x^T\beta_k}}{\sum_{l=1}^{K}\exp^{\beta_{0,l}+x^T\beta_l}}, \tag{4}$$

and observations are assigned to classes based on whichever class has the highest associated probability.

Once we have the probabilities, assigning individuals to a class is quite simple, however, the process of determining these probabilities is not a straightforward task. Using standard maximum likelihood for this purpose, without any constraints, would lead to the situation where any coefficients of the form $\{\beta_{0,l}, \beta_l\}_1^K$ and $\{\beta_{0,l} - c_0, \beta_l - c\}_1^K$ result in equal probabilities, as they would cancel out in Equation 4. In addition, multinomial logistic regression is not able to deal with substantial autocorrelation, which is in no way prevented here. To solve both of these problems, the multinomial logistic regression is used in combination with the elastic-net penalty.

Regularisation is commonly applied to regression techniques to improve prediction accuracy as well as interpretation. It generally involves including a penalty in the objective function to either reduce or eliminate particular coefficients, as in general, it holds that including all variables may lead to overfitting (and subsequently poor performance) and may weaken the interpretability of all that is measured (Tibshirani, 1996). Elastic-net is a combination of two regularisation techniques, namely ridge regression and the lasso, which approach regularisation from two different angles (Friedman et al., 2010). Ridge regression, applies continuous shrinkage to the coefficients based on a bias-variance tradeoff (Zou and Hastie, 2005). The lasso, on the other hand, is a penalised least-squares regression technique which also applies continuous shrinkage, but does so in combination with variable selection. Whereas Ridge regression shrinks the coefficients of correlated regressors, the lasso method picks one and ignores the rest. In contrast to the lasso method, Ridge regression is ideal if the data has many regressors which are all non-zero, however, it does not lead to a parsimonious model. This is because shrinking coefficients is not equivalent to removing them, leading to a potentially excessive number of parameters. By combining the two, elastic-net leads to a parsimonious model with better predictive power. In addition, the elastic-net shows the grouping effect, which implies that highly correlated variables will receive similar coefficients.

The penalty incorporated in elastic-net is a convex combination of the ridge regression penalty and the lasso penalty. Ridge regression minimises a penalised sum of squares using an L_2 norm, as shown in Equation 5, and the lasso minimises a penalised sum of squares including an L_1-penalty, illustrated in Equation 6.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda_1 \sum_{j=1}^{p} \beta_j^2 \tag{5}$$

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda_2 \sum_{j=1}^{p} |\beta_j| \tag{6}$$

Setting α equal to $\frac{\lambda_1}{\lambda_1+\lambda_2}$, the elastic-net penalty becomes $(1-\alpha)\sum_{j=1}^{p}|\beta_j| + \alpha\sum_{j=1}^{p}\beta_j^2$. When $\alpha = 0$, the model reduces to the ridge regression and similarly, when $\alpha = 1$ the model

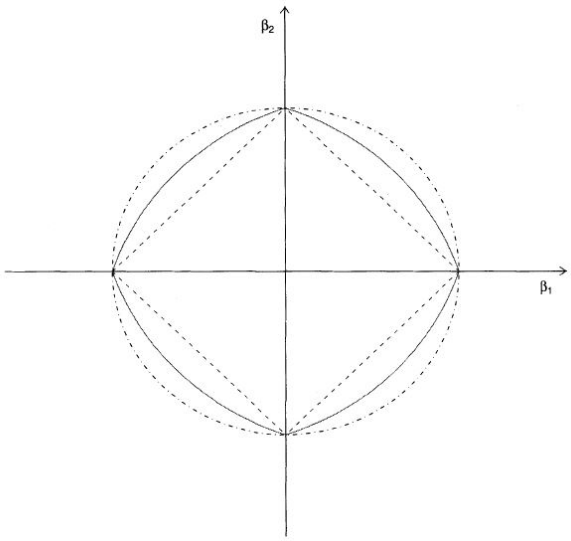


Figure 2: A two-dimensional contour-plot where the outer circle represents the ridge penalty, the inner square represents the lasso penalty and the shape in between represents the elastic-net penalty (Zou and Hastie, 2005).

is equivalent to the lasso. The convexity of the elastic-net penalty, as well as the difference in structure of the ridge and lasso penalties is illustrated in Figure 2.

The remaining question is how to determine the optimal balance between the two penalties. Using the R *glmnet package*, this is done sequentially (Simon et al., 2011). First, the *cv.glmnet* function is run for multiple α values, each of them with a prespecified vector of one hundred λ values. The best value for α is that which leads to the lowest average Area Under the Receiver Operating Curve (AUROC, see Section 4.5.2, where the average is taken over the three folds in which the data is split (see Section 3), as well as an additional three folds provided as input in *cv.glmnet*. Based on this particular α, the λ is set to its largest possible value such that the multinomial deviation is at most one standard error larger than its minimum. Setting λ in this manner is common practice as it ensures that the model is sparse, whilst still achieving a similar degree of accuracy. Having determined the penalty parameters, the next step is to solve the penalised negative log likelihood. In order to do so, the Newton algorithm in combination with partial Newton steps is used. For more details on the exact execution, see the work by Friedman et al. (2010).

4.2 Gradient Boosting

Boosting is an ensemble method which efficiently combines the output of many weak learners, also called base learners, to provide a more accurate overall prediction. These weak learners, typically simple decision trees, have only one requirement, namely to outperform random guessing, but by combining them intelligently, their variability can be exploited to obtain robust and precise results (Schapire, 1990).

4.2.1 Decision Trees

The basis of boosting lies with the weak classifiers it is built on, in this case decision trees. A decision tree, as presented by Breiman et al. (1984) makes predictions by splitting observations into groups based on certain attributes, and has become a staple in classification problems. By making splits iteratively, decision trees do not focus on the effect of one variable in particular, but rather on the combination of variables which lead to a decision. Making splits is equivalent to making subsets of your data, based on a list of questions, until these subsets can be assigned a label. In that sense, decision trees are in fact all about asking the right questions.

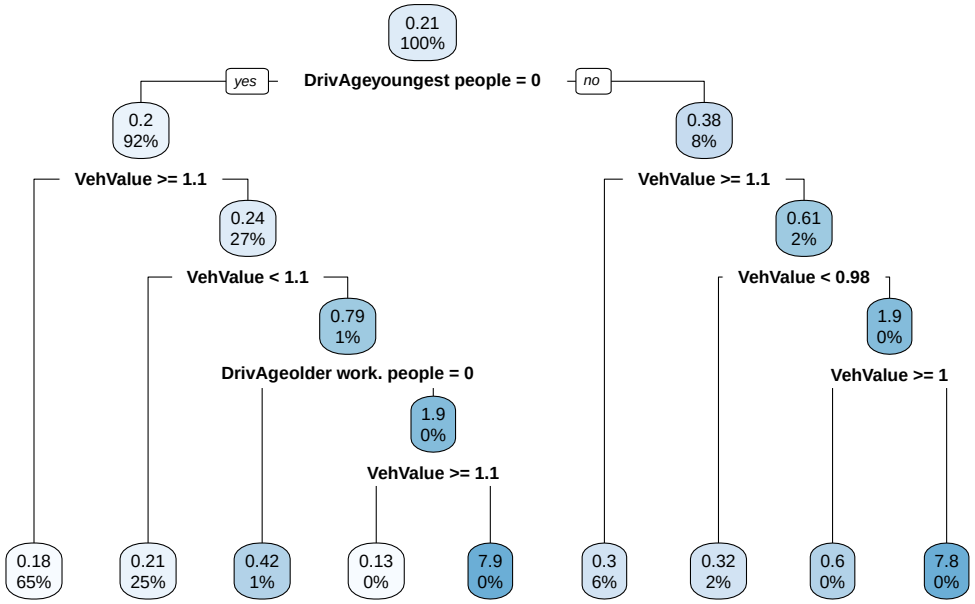


Figure 3: A binary decision tree based on the *ausprivauto0405* dataset, containing claims from third party automobile insurance in Australia, from the *CASdatasets* package (Dutang, 2016). The tree was generated using the *rpart* package and plotted using *rpart.plot* (Therneau and Atkinson, 2018; Milborrow, 2018).

An example of a decision tree can be seen in Figure 3, which aims to identify how often someone will submit a claim for their third party automobile insurance. The tree starts at the top node, referred to as the root node, and initially splits the data based on whether or not the driver does not belong to the category of youngest people. The group on the left does not belong to this group, i.e. *DrivAgeyoungest people = 0*, whereas the group on the right does, i.e. *DrivAgeyoungest people = 1*. The data is again divided at each subsequent node, using the vehicle value, i.e. *VehValue* first in both the left and right branch, until ending at one of the nine leaves (the final nodes). Each of these leaves is then assigned the average number of claims in that leaf as the expected number of claims submitted. This example contains only numerical (VehValue) and dummy (DrivAge) variables. However, decision trees can also be constructed using categorical variables without first converting

them into dummies.

Although the concept of a decision tree is quite simple, its implementation raises a few questions, the first being how to determine the splits. The aim of the tree is to continue splitting the observations until the level of impurity, i.e. the degree with which classes are mixed at the nodes, is minimised. The first split should then use the variable which leads to the largest decrease in impurity. There are three commonly used measures of impurity, namely the Gini index, the level of entropy and the classification error and they are defined as follows:

$$Gini(t) = 1 - \sum_{i=1}^{C} p(i|t)^2 \tag{7}$$

$$Entropy(t) = -\sum_{i=1}^{C} p(i|t) \log_2 p(i|t) \tag{8}$$

$$ClassificationError(t) = 1 - \max_i p(i|t) \tag{9}$$

where i is one of the C classes and t is a particular node. This thesis will make use of the Gini index because it is less computationally intensive than entropy, and more sensitive to changes in class probabilities than the classification error. Furthermore, the difference in accuracy between these methods is found to be somewhat negligible (Mingers, 1989).

Another question to consider is when to stop splitting. If no restrictions are placed on the frequency of splits, then a decision tree will be prone to overfitting, continuing to split until it can identify each individual label. Such specificity makes it hard to generalise results and hence is an undesirable outcome. To prevent this, various splitting criteria have been introduced; stop when all observations in a node belong to the same class, stop when all observations in a node have similar attributes and stop when the number of observations in a node is less than a pre-specified threshold. In addition, there is the option to perform cost-complexity pruning on the decision tree. This process removes the subtrees[5] which do not significantly reduce the error rate over the number of nodes, i.e. cost-complexity (Breiman et al., 1984). Pruning can either be done after having built the tree, or during the process of building one. The latter would prevent a tree form introducing new splits if these did not exceed the given cost-complexity parameter.

Whereas a decision tree has a simple implementation and can provide clear insights, it has some downsides. For one, a decision tree is prone to high variability depending on the observations and variables selected. Although this also holds for other methods, this is especially true for decision trees as the first split may be rather decisive for the overall outcome. Secondly, the results are highly dependent on the control parameters, creating a trade-off between accuracy and overfitting. Nevertheless, by incorporating these trees in an ensemble method, we can significantly reduce the instability and variability of the results and simultaneously achieve superior outcomes.

4.2.2 Gradient Boosting

The main idea behind boosting algorithms is that of iterative adjustment using simple learners. More specifically, during each iteration a simple learner is trained and the observation

[5] A subtree is a tree which forms a subset of another tree.

weights are adjusted based on the mistakes made in the previous round. In the case of decision trees, this means that we build a tree, examine the mistakes made and increase the weights of the corresponding observations. We then build another tree using this adjusted weights and the process continues until the maximum number of iterations has been reached or the increase in performance does not exceed a prespecified threshold.

The original implementation of boosting, namely AdaBoost, as developed by Freund et al. (1996), has seen an immense increase in popularity and diversification over the past two decades. This method was originally only able to deal with very specific loss functions, but has since been adapted and generalised in various ways, the most significant being the adaptation referred to as gradient boosting (Friedman, 2000). This method is based on the concept of gradient descent and allows for a larger variety of loss functions as well as regression problems and multi-class dependent variables. Albeit that multi-class versions of AdaBoost also exist, these methods either have limited implementability or are based on a one-vs-one or one-vs-all construct, where rather than jointly optimising the problem, a multitude of binary models are combined (Freund et al., 1996; Appel et al., 2016; Zhu et al., 2009). For this reason, gradient boosting is deemed more appropriate for this application.

The reason that boosting, along with other ensemble methods, has become so successful, is its ability to combine simple classifiers, e.g. decision trees, in order to generate highly accurate results. Moreover, tree-based gradient boosting is able to handle both regression and classification problems whilst requiring little to no data preprocessing as well as parameter tuning (Guelman, 2012). This method also circumvents the highly debated issue of dealing with missing values as it can easily incorporate these unknowns as a class of their own. The advantage of this is that it can, to a large degree, remove subjectivity from the methodology.

Gradient boosting distinguishes itself from other ensemble methods by having a weight-based iterative procedure. Gradient boosting starts with equal observation weights and then generates a decision tree with the aim of correctly segregating observations such that each group can be assigned to a class. The weights of the misclassified observations in this initial tree are then increased, in order to emphasise these observations in the subsequent tree. Another decision tree is generated, and again the observations which have been incorrectly classified obtain larger weights. This process continues until no (significant) improvement can be made, or until the number of pre-specified tree has been reached. Formally, the algorithm for K-class gradient boosting is such as is presented in Algorithm 1, where $\tilde{y}_{ik}$ are the derivatives of the loss function and γ_{ik} are the model updates for each iteration.

Algorithm 1 K-Class Gradient Boosting

$F_{k0}(\mathbf{x}) = 0, \quad k = 1, K$
for $m = 1$ to M **do**
  $p_k(\mathbf{x}) = \exp(F_k(\mathbf{x}))/\sum_{l=1}^{K} \exp(F_l\mathbf{x})), \quad k = 1, K$
  for $k = 1$ to K **do**
    $\tilde{y}_{ik} = y_{ik} - p_k(\mathbf{x}_i), \quad i = 1, N$
    $\{R_{jkm}\}_{j=1}^{J} = J -$ terminal node $tree(\{\hat{y}_{ik}, \mathbf{x}_i\}_1^N)$
    $\gamma_{jkm} = \frac{K-1}{K} \frac{\sum_{\mathbf{x}_i \in R_{jkm}} \tilde{y}_{ik}}{\sum_{\mathbf{x}_i \in R_{jkm}} |\tilde{y}_i k|(1-|\tilde{y}_{ik}|)}, \quad j = 1, J$
    $F_{km}(\mathbf{x}) = F_{k,m-1}(\mathbf{x}) + \sum_{j=1}^{J} \gamma_{jkm}\ \mathbf{1}(\mathbf{x} \in R_{jkm})$

Although the amount of necessary parameter tuning is quite minimal, some parameters

must still be determined. For gradient boosting these parameters are the number of trees, the complexity of the trees and the learning rate. The number of trees, i.e. the number of iterations, can simply be set to a very large number because at some point the weights will converge. However, this is not the most efficient approach in terms of computation time. In practice, one can plot the development of the accuracy level as the number of trees is increased and set the number of trees to that where the accuracy level stabilises. With regard to complexity, the interaction depth of the decision trees, equal to the number of splitting nodes, is considered. Increasing the interaction depth increases the degree of precision, however, it simultaneously increases the computational complexity as well as the risk of overfitting. Nevertheless, by increasing the number of trees, the severity of this trade-off is subdued. Figure 4 presents an example of tuning as generated by the R *caret* package (Kuhn et al., 2017) for the same *ausprivauto0405* dataset as in Figure 3. Boosting is applied to one hundred bootstraps of the insurance data and the average performance of these bootstraps is presented for a range of values for the number of trees as well as interaction depth. The optimal number of trees is quite subjective in this case, however, from 600 trees onwards, very little progress is made in absolute terms. The best depth is more straightforward, in this case being seven, as this curve is consistently above the rest.

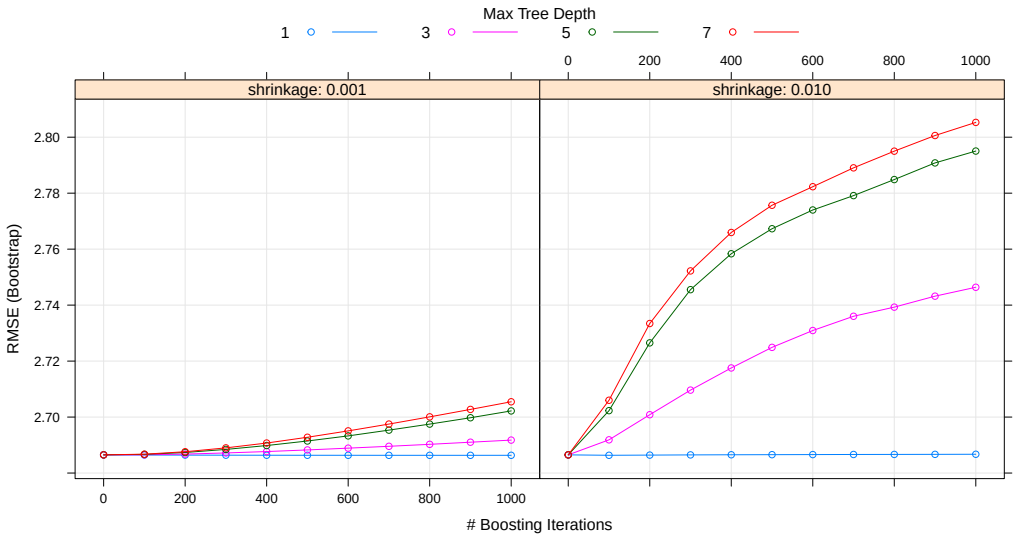


Figure 4: Output of the *caret* package's train function to tune a gradient boosting machine on the *ausprivauto0405* dataset from the *CASdatasets* package (Dutang, 2016).

The third parameter to be determined is the learning rate, denoted by ν. This rate is in fact a regularisation measure, and hence limits the extent of overfitting. By restricting the fitting procedure to some extent, one can maintain generalisability of the outcomes (Friedman, 2000). Algorithm 1 can be easily adjusted to incorporate the learning rate by replacing line 8 with the following:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot \rho_m \cdot h(\mathbf{x}; \mathbf{a}_m), \qquad 0 < \nu \leq 1. \tag{10}$$

The optimal learning parameter can be determined similarly to the maximum tree depth by

evaluating the performance of these parameters using 3-fold cross-validation in each training set. Multiple iterations of gradient boosting are applied to a range of values for the learning rate. The best learning rate is that which results in the highest overall performance on the test set(s). Figure 4 also compares two values of the learning rate, namely 0.001 and 0.01, with the latter clearly outperforming the former.

Once the parameters have been determined, there are several options for determining the impact a particular variable has on the outcomes. The first such option is the measure of relative influence, as first introduced by Breiman (2001). Roughly speaking, this measure determines the average empirical improvement from splitting on a particular variable, across all trees used in the gradient boosting algorithm. This improvement is typically measured in terms of the variable influence, as denoted in Equation 11, where $Influence_j(T)$ is the influence of variable j in a particular tree T with L splits (Natekin and Knoll, 2013).

$$Influence_j = \sum_{i=1}^{M} Influence_j(T_i) \tag{11}$$

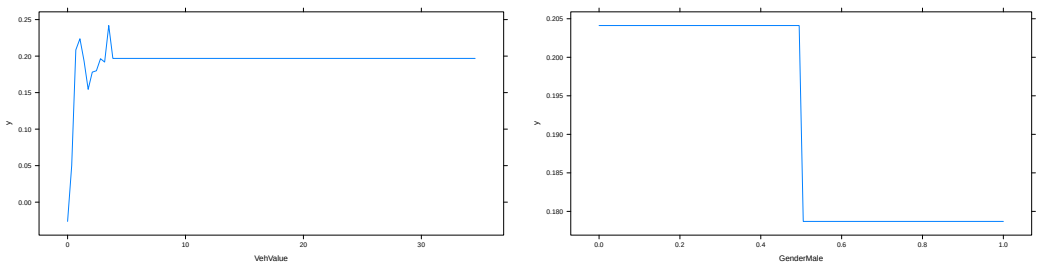
where

$$Influence_j(T) = \sum_{k=1}^{L-1} I_k^2 \cdot 1(S_k = j). \tag{12}$$

Moreover, this influence measure is determined by the frequency with which the variable j is selected, as seen by the indicator function, as well as the squared improvement as a result of including this split. The latter is based on the respective objective function.

The relative influence of variables leads to a ranking of their respective importances. However, this is merely a relative measure. A more sophisticated approach, which allows us to gain more insights into these effects, is a partial dependence plot. Such a plot can illustrate the effect a particular variable has on the outcome classes over the domain of that variable. This could be done by taking the partial derivative of the objective function with respect to that variable, for each class. However, especially in the case of big data, this becomes too computationally intensive. Therefore, in accordance with the work by Friedman (2000) this can also be achieved by setting the remaining variables to their respective sample means and modes (Natekin and Knoll, 2013).

Using the automobile insurance data offered by Dutang (2016), Figure 5 shows two examples of such partial dependence plots, demonstrating the measured effects of vehicle value and gender on the number of claims made. In Figure 5a, a continuous and non-linear relationship between the dependent and independent variable is found, without having to transform either of the variables. Given that gender is a binary variable (in this case, whether or not the individual is male), Figure 5b depicts that men have a lower probability of submitting a claim than women, in this sample. These plots prove that, in terms of interpretation and flexibility, gradient boosting has a clear advantage as compared to GLM and neural networks.



(a) A partial dependence plot of vehicle value.

(b) A partial dependence plot of gender.

Figure 5: Examples of partial dependence plots of vehicle value and gender on the number of claims made resulting from the *gbm* package (Ridgeway and with contributions from others, 2017). The data used is the *ausprivauto0405* dataset from the *CASdatasets* package (Dutang, 2016).

Lastly, this thesis makes use of the *gbm* R package to implement gradient boosting (Ridgeway and with contributions from others, 2017).

4.3 Neural Networks

Despite having been around since the 1940's (Wasserman and Schwartz, 1988), neural networks have only really gained popularity in the past decade. The reasons for this development are two-fold, namely the ability of these methods to provide highly accurate predictions and the improvements made in computation power, allowing for a much swifter implementation. This method, based on the workings of the human brain (hence the name), has proven its worth specifically in image recognition by winning the ImageNet Large-Scale Visual Recognition Challenge in 2012 (Russakovsky et al., 2015). Even so, the application of neural networks in the business sector has been quite limited. The main reason for this is that, regardless of their accuracy, neural networks are considered to be a black box. Although this may be true to some extent, there are various ways to circumvent this issue and still obtain insights from the results of a neural network. These methods will be discussed in this section, but first the exact workings of a neural network are explained in more detail. Note that this thesis is considering the standard feedforward neural networks with one hidden layer. In the case of supervised learning, these networks are also referred to as Multilayer Perceptrons (MLP).

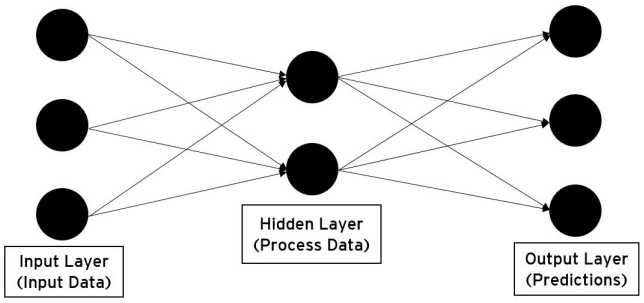


Figure 6: A simple illustration of a feedforward neural network with one hidden layer.

As mentioned briefly, neural networks are designed similarly to the human brain (though are of course more simple in construction), and hence a lot of the terminology related to neural networks stems from biology. The general idea is that the set of variables in the data are processed by a hidden layer which is then linked to the output nodes which represent the assigned outcomes (Francis, 2001). This idea is illustrated in Figure 6. As can be seen, each independent variable, represented by an input node, is connected to each node in the hidden layer. These nodes are in turn also connected to each of the output nodes. The role of the hidden layer is to approximate complex functions in the data by applying non-linear activation functions, and hence creates the relations between input and output. The more nodes, also called neurons, included in this layer, the more complex these relations can become. In addition, the role of these activation functions is to iteratively adjust the strength of the signals between the connected nodes, with the aim of increasing performance. This idea of increasing or decreasing the weights based on mistakes made in the previous round is what brings about the learning aspect of neural networks and furthermore, is a concept which is quite similar to that of boosting.

As per usual, several choices with respect to the model settings need to be discussed. Neural networks, being a rather complex method, have the risk of not converging, hence the model inputs must be chosen wisely. Firstly, since the dependent variable is categorical, the most appropriate activation function is the softmax activation function (Bishop, 1995). This function is a generalisation of the logistic function which forces the outcomes to lie between 0 and 1, i.e. in the form of probabilities. The softmax activation function is as follows:

$$y_k = \frac{1}{1 + exp(-A_k)} \tag{13}$$

where

$$A_k = a_k - \ln\Big[\sum_{k' \neq k} exp(a_{k'})\Big] \tag{14}$$

and where a_k is derived from the conditional probability of belonging to class k, given a unit activation vector $\mathbf{z}$ (for more details see Bishop (1995)).

Before applying this activation function however, the neural network requires an initial set of weights to start its computation process. These initial sets are key to the overall performance in the model and setting them randomly leads to poor results, as is demonstrated in the work by Glorot and Bengio (2010). A more sophisticated approach, accounting for the gradients of the activation function, is the Xavier initialisation. Improper, i.e. random or small, initial weights may lead to derivatives equal to zero, causing the optimisation process to fail. In order to maintain the variability of the derivatives, the Xavier initialisation dictates using initial weights which are drawn from $N\big(0, \sqrt{\frac{1}{n}}\big)$ where N is the normal distribution. The idea behind this is based on the relation between the variance of the different layers. Let x_j^i be one of n nodes in layer i of a neural network, and x_k^{i+1} be a node in layer $i+1$. Between these layers is a series of weights and hence x_k^{i+1} can be expressed as a linear combination of the nodes of the previous layer and the weights (w_{jk}^i) connecting them to this k^{th} node: $x_k^{i+1} = \sum_{j=1}^n w_{jk}^i x_j^i$. Aggregating over all nodes in layer $i+1$ and taking the variance in both sides gives us $Var(x^{i+1}) = \sum_{j=1}^n Var(x^i)Var(w^i)$. Note that we assume here that the means of the weights are equal to zero. Given that we want to maintain the variability of the nodes throughout the layers of the neural network, we can impose the

relation that $Var(x^{i+1}) = Var(x^i)$ and it follows that $Var(w^i) = \frac{1}{n}$. In a one-layer neural network n is equal to the number of input variables provided.

Although Xavier initialisation has its benefits, it has one major disadvantage, namely, dealing with non-linearities. He et al. (2015) proof this in their research, and provide a simple adjustment to the Xavier weights by drawing them from a $N\big(0, \sqrt{\frac{2}{n}}\big)$ distribution instead. The neural networks discussed in this thesis have initial sets which are drawn from this distribution.

Another step to aid convergence is data preparation. It is commonly advised to first standardise the data, although there is little consensus with regards to the exact procedures for doing so (Dawson and Wilby, 1998). However, Shanker et al. (1996) find that statistical standardisation tends to outperform linear transformation (also known as min-max tranformation [6]). In addition, non-numerical variables should be, to some extent, centred around zero (S. Sarle, 2002). Binary variables should hence use the values $-1, 1$ rather than $0, 1$ and categorical variables should be enumerated by means of effects coding, also known as C-1 coding. Effects coding is different from the default means of expanding a categorical variable because rather than making a binary variable for each of the C levels of this variable, only $C - 1$ dummies are made. When an observation has the C^{th} level, all dummies take the value -1. An example is provided in Table 3 where the observations take levels A, B, C and D (which is excluded) respectively. In addition, to prevent overfitting, a small weight decay of 0.000001 is used, as a small decay has shown to improve out of sample performance substantially (Krogh and Hertz, 1992). Lastly, a bias unit, equivalent to the intercept in standard regression models, is included in the model. For the implementation of this method the *nnet* R package as developed by Venables and Ripley (2002) is used.

Table 3: Example of Effects Coding

	Level A	**Level B**	**Level C**	**Original Variable**
Observation 1	1	0	0	A
Observation 2	0	1	0	B
Observation 3	0	0	1	C
Observation 4	-1	-1	-1	D

Similarly to gradient boosting, neural networks are often resented for the lack of insights they provide. Although the computed weights do provide an indication of the importance of a particular variable, it is not feasible to derive the precise effect of one variable from all these weight combinations, even in a simple model such as that presented in Figure 6. However, Francis (2001) came up with a straightforward suggestion to circumvent this issue. In his work on neural networks, he defined the sensitivity of a particular variable to be its contribution towards decreasing the mean squared error on the test set. This is measured by comparing the forecast performance to that when the specified variable is fixed. This concept of sensitivity is not only straightforward, but most importantly (given the computational intensity of neural networks), has the advantage of not requiring to retrain the networks. This thesis uses a slightly altered version of this sensitivity, however, by considering the contribution of a variable towards the average Area Under the Receiver Operating Curve (AUROC, see Section 4.5.2) rather than the mean squared error.

[6] $x* := \frac{x - x_{min}}{x_{max} - x_{min}}$

Standard methods, such as those that categorise as a GLM, have many appealing statistical properties and hence provide much deeper insights with regard to the relationships found between variables. In addition, these methods perform well when the relations between the dependent and independent variables are linear. However, many datasets contain complex and non-linear relations which cannot be captured by a simple linear construction and require the use of advanced data-mining techniques. The techniques chosen in this research, boosting and neural networks have proven their worth when applied to substantial datasets and in particular when predicting rare instances due to them being universal function approximators (Friedman, 2000; Francis, 2001).

4.3.1 Other Methods

Of course, boosting and neural networks are not the only available data-mining techniques, however they were deemed most appropriate for this application. Random forests and bagging are similar to boosting in the sense that they are also tree-based ensemble methods, however, whereas boosting iteratively adjusts the weights it assigns to its observations, random forests and bagging iteratively build trees based on bootstraps of the sample (Liaw et al., 2002; Breiman, 1996). This is less effective in the context of imbalanced data, because each bootstrap will have a similar level of imbalance and hence the individual trees will still struggle to identify the minority classes. Figure 7 shows the distribution of the proportion of majority class observations in one million bootstraps. In the original sample the majority class consists of 92% of the population and the bootstrap proportions hardly deviate from this level. However, you could somewhat circumvent this problem by a priori balancing the data.

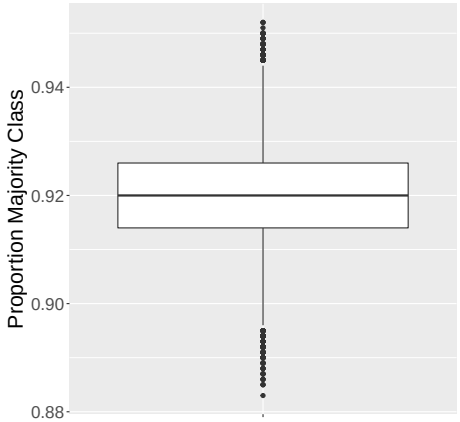


Figure 7: Boxplot of the proportion of majority class observations in one million bootstraps. The proportion in the original sample is 92%.

Another method which is commonly applied to imbalanced data is that of Support Vector Machines (SVM). Whereas this method has proven to be effective in detecting extreme observations by means of splines, it does so in the case of binary data (Hsu and Lin, 2002). Possible alterations for multi-class data have been provided, but these methods either fail in terms of feasibility or in terms of performance. Due to the lack of consensus as well as

refinement for this application, SVMs are not considered. Any remaining techniques have previously been outperformed or are deemed outside of the scope of this research.

4.4 Dealing with Imbalanced Data

When data is imbalanced it means that there is an uneven distribution of observations amongst the classes. The most common example is that of binary data where the majority of the data belongs to one class, denoted as the majority class, and where only a small selection belongs to the remaining minority class. Multi-class imbalanced data, where there may be several majority or minority classes, is also possible, but has been a far less popular field of research. However, this thesis will focus on data which contains three classes, two of which being minority classes and the smallest of all being of particular interest.

The reason that imbalanced data is a problem is due to its impact in the field of classification, in particular in machine learning. Machine learning techniques assume a balanced distribution and are strongly accuracy-based. The issue with assuming a balanced distribution is that the methods cannot emphasise one class and they put an equal weight on each occurring. This may defeat the purpose in many scenarios, where the aim of the research is to identify the minority. Furthermore, accuracy only looks at the frequency with which classes are identified correctly, without distinguishing between the classes. In order to illustrate the severity of these problems, consider the example where there is data on 1000 patients, of which ten have a fatal condition. We are interested in predicting which patients have this rare disease but, without making any adjustments, standard machine learning techniques are likely to predict that every patient is healthy, simply because odds are that this is the case. This will lead to an accuracy of 99% which is quite exceptional, yet completely useless for the investigation. For this reason, either the input or the implementation of machine learning methods need to be adjusted in order to focus on predicting the class of interest. Methods for doing so are discussed below.

4.4.1 Over- and Under-Sampling

The simplest way of dealing with imbalanced data is to make the data balanced. Depending on the data, this can be done by either under-sampling the majority class(es) or over-sampling the minority class(es). If your data is rather substantial, then under-sampling is the appropriate option. In this case you would select a subset of your majority class and discard the rest from your investigation. There are various methods to select this subset, the easiest being to pick the observations at random. The disadvantage of selecting the observations at random, however, is that there is no guarantee that the subset is representative of the complete sample. For this reason, Mani and Zhang (2003) came up with a more sophisticated approach, based on the average and maximum distances of particular observations to the minority. However, the performance of this method is quite disappointing and it is not very practical for large datasets. Another method, which involves clustering the majority class prior to generating the subsample, will be discussed in more detail in Section 4.4.3. If your dataset is not that large, then under-sampling would not be wise, as the information in the data would become even more limited. In this case one could over-sample the minority class instead. This too can be done at random, by randomly selecting observations from the minority class and adding an exact copy of this observation to the data until the desired number of minority observations is reached. The downside of this method is that it leads to an overly specific decision region (due to low variability) and consequently, overfitting

(Chawla et al., 2002). To overcome this issue, other methods have been developed, the most popular being the Synthetic Majority Over-Sampling Technique (SMOTE), which is discussed in Section 4.4.2. When it comes to over- and under-sampling, the main question at hand it to which extent the data should be sampled. Sampling until the data is exactly balanced seems quite excessive, resulting in a dataset which has either shrunken substantially or has a multitude of exact copies of the minority class. Consequently, most research examines the results for a variety of ratios and evaluates the performance of methods over the range of ratios considered.

4.4.2 Multi-Minority SMOTE

The original version of the SMOTE technique, as developed by Chawla et al. (2002), was designed to both undersample as well as oversample the data, where oversampling is done by creating synthetic observations rather than exact copies. These observations, made through a combination of bootstrapping and k-Nearest Neighbours (kNN), are very similar to the original ones, but due to their differences avoid overfitting the model. For a particular observation, the kNN method finds the k observations which are closest to it based on an overall distance measure. In the case where all variables are numeric, various distance measures are available such as the Euclidean, Manhattan and Minkowski distances. However, in order to allow for nominal variables, the Value Difference Metric by Cost and Salzberg (1993) is used. This metric considers the overlap of any two feature values V_1 and V_2 amongst the classes. More formally, the distance measured is expressed as:

$$\delta(V_1, V_2) = \sum_{i=1}^{n} \left| \frac{C_{1i}}{C_1} - \frac{C_{2i}}{C_2} \right|^k \tag{15}$$

where C_1 is the total frequency of feature V_1 and C_{1i} is the frequency of V_1 in class i. The same relation holds between C_2, C_{2i} and V_2. The value k is a constant which is usually set to 1. The weighted distance between two feature vectors X and Y can then be expressed as:

$$\Delta(X, Y) = w_x w_y \sum_{i=1}^{N} \delta(x_i, y_i)^r \tag{16}$$

where x_i and y_i are the features in X and Y, and where w_x and w_y are the exemplar weights. For a new feature vector, w_y is set to 1 whereas w_x is the ratio of the total uses of a feature vector over the total number of correct uses of that vector and hence depicts the bias towards reliable examples. When $r = 1$, Equation 16 yields the Manhattan distance and it yields the Euclidean when $r = 2$. Synthetic observations are then generated at random in the space between the observation in consideration and its selected nearest neighbour. For more details on the specific features of kNN see Cost and Salzberg (1993).

The exact steps of SMOTE are presented in the pseudo code in Figure 8. This method has been applied vigorously since its introduction and has become one of the most popular techniques for dealing with imbalanced data. Adjusted versions of SMOTE have also been proposed, but these methods are often designed for a specific application, for example when there are distinct borders between the majority and minority samples (Han et al., 2005b).

Consequently, there is not yet (at the date of this research) a version of SMOTE which can handle multiple minority classes. This thesis presents an alteration of the original SMOTE, dubbed Multi-Minority SMOTE (MM SMOTE) which can balance data with

Algorithm *SMOTE*(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE $N\%$; Number of nearest neighbors k
Output: $(N/100)$ * T synthetic minority class samples
1. (∗ *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* ∗)
2. **if** $N < 100$
3. **then** Randomize the T minority class samples
4. $T = (N/100) * T$
5. $N = 100$
6. **endif**
7. $N = (int)(N/100)$ (∗ *The amount of SMOTE is assumed to be in integral multiples of 100.* ∗)
8. k = Number of nearest neighbors
9. $numattrs$ = Number of attributes
10. $Sample[\,][\,]$: array for original minority class samples
11. $newindex$: keeps a count of number of synthetic samples generated, initialized to 0
12. $Synthetic[\,][\,]$: array for synthetic samples
(∗ *Compute k nearest neighbors for each minority class sample only.* ∗)
13. **for** $i \leftarrow 1$ **to** T
14. Compute k nearest neighbors for i, and save the indices in the $nnarray$
15. Populate(N, i, $nnarray$)
16. **endfor**

Populate(N, i, $nnarray$) (∗ *Function to generate the synthetic samples.* ∗)
17. **while** $N \neq 0$
18. Choose a random number between 1 and k, call it nn. This step chooses one of the k nearest neighbors of i.
19. **for** $attr \leftarrow 1$ **to** $numattrs$
20. Compute: $dif = Sample[nnarray[nn]][attr] - Sample[i][attr]$
21. Compute: gap = random number between 0 and 1
22. $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
23. **endfor**
24. $newindex{+}{+}$
25. $N = N - 1$
26. **endwhile**
27. **return** (∗ *End of Populate.* ∗)
End of Pseudo-Code.

Figure 8: Pseudocode for the SMOTE algorithm

multiple minority classes, in this case being both the medium and high risk profiles. The process of undersampling the majority class remains the same, however, the minority classes are oversampled to a degree which is inversely proportional to their relative frequencies. Applying this technique hence results in a dataset in which all classes are balanced.

4.4.3 Cluster-Based Under-Sampling

This method aims to find a representative subsample of the majority class by means of clustering, as proposed by Yen and Lee (2009). The idea is that by first clustering the data, K relatively homogeneous subgroups can be identified. If the bulk of the sample comes from the majority class, then this sample is thought to behave as a majority sample and the same applies vice versa for the minority class. If the ratio between the majority and the minority class is $m:1$ then a subsample of majority class observations of size $SSize_{MA}$ is extracted from each cluster i, where $1 \leq i \leq K$. $SSize_{MA}$ is defined as follows:

$$SSize^i_{MA} = (m \cdot size_{MI}) \cdot \frac{size^i_{MA}/size^i_{MI}}{\sum_{i=1}^{K} size^i_{MI}/size^i_{MI}} \tag{17}$$

where $size^i_{MA}$ is the number of majority class samples in cluster i, and $size^i_{MI}$ is the number of minority class samples in cluster i, which is assumed to be at least one. The majority class observations extracted from the clusters are then combined with the entirety of the minority sample. The result is a balanced dataset which can be used for classification.

Although this method has the advantage of producing a representative sample of majority class observations, it brings about its own downsides. The first is that there is no indication of which clustering technique should be used nor how many clusters should be constructed. This makes it difficult to generalise the method and moreover, makes the results highly dependent on the clustering technique chosen. K-Means, for example, despite being a popular choice, forces clusters to be similar in size and has no objective measure to determine the optimal number of clusters to be used. Moreover, the disadvantages of over-sampling still apply, namely that a large chunk of information is discarded and that the dataset size has been significantly reduced.

4.4.4 Cost-Sensitive Methods

Contrary to the previous methods, which were all data-based methods, cost-sensitive methods are algorithm-based techniques for dealing with imbalanced data. Whereas data-based methods focus on adjusting the dataset to make it more balanced, algorithm-based methods focus on adjusting the methods themselves such that they prioritise the minority class. As discussed previously, machine learning techniques are accuracy-based and do not distinguish between a mistake in predicting the majority class and a mistake in the minority class. However, by assigning costs to the different types of mistakes, you can adjust the behaviour of the technique. Considering the example of disease diagnosis, a Type I error, i.e. a false negative, where a patient is classified as healthy whilst he is in fact a carrier of the disease is the most costly error a method can make. A Type II error, classifying a patient as ill whilst he is healthy, is also unfortunate, but can easily be corrected by testing the patient. The costs for performing the test can be easily determined, excluding any costs of stress resulting from a misdiagnosis, but the cost of not detecting a disease is not as easy to pinpoint, seeing as this is not a one-dimensional monetary amount. This inability to clearly determine the

costs of errors is what makes these methods less applicable than data-based ones. However, in the field of car insurance, one can use historical claims as a proxy for these costs.

In the boosting algorithm, cost-sensitivity can be introduced by adjusting the loss function. Although many methods have been presented to implement this procedure, most are, again, based on binary boosting or depend on Bayesian techniques to determine conditional probabilities. The prior can lead to sub-optimal decisions (as they do not jointly-optimise the problem) and do not allow for asymmetrical cost matrices. The latter, on the other hand, has the difficulty of determining accurate conditional probabilities. Circumventing this problem, as well as addressing a new problem called guess-aversion, is the work by Beijbom et al. (2014). In their paper, they present several guess-averse, cost-sensitive adaptations of gradient boosting, where guess-averse implies that these functions promote correct classification rather than arbitrary guessing in the case of equal scores. Of the functions they presented, the Generalized Logistic Loss (GLL) proved to be most successful (Beijbom et al., 2014).

Considering the generic case with M classes, a cost matrix C and an observation x of class z, the GLL can be expressed as:

$$L^{log,exp}(C, z, S(x)) = \log\left(1 + \sum_{j=1}^{M} C_{z,j} \exp^{S_j(x) - S_z(x))}\right) \tag{18}$$

where $C_{z,j}$ is the cost of assigning an observation of class z to class j and $S_j(x)$ is the confidence with which observation x can be assigned to class j. In the case of cost-insensitive classification, this method is equivalent to that presented in Section 4.2.2.

Neural networks also typically consider the number of errors, rather than the nature of the errors. However, this too can be adjusted by incorporating the associated cost matrix into the loss function. Kukar et al. (1998) investigate four different loss functions for back-propagation neural networks, namely, cost-sensitive classification, adaptive output, adaptive learning rate and minimisation of misclassification costs. Having tested these methods on 11 different UCI datasets, the latter proved to be most successful. In order to implement this method, first a cost matrix as well as a cost vector need to be defined:

$$CostVector[i] = \frac{1}{1 - P(i)} \sum_{j \neq i} P(j) \cdot C[i, j] \tag{19}$$

where $P(i)$ is the prior probability that an observation belongs to class i. The minimisation of misclassification costs also requires a factor K which corrects the error function, but simultaneously retains the behaviour of the neural network when the cost matrix is uniform. This factor is denoted by:

$$K[i, j] = \begin{cases} CostVector[i], & i = j \\ C[i, j], & i \neq j \end{cases} \tag{20}$$

and the associated loss function is then:

$$L = \sum_{p \in Examples} \frac{1}{2} \sum_{i \in Output} ((y_i - o_i) \cdot K[class(p), i])^2. \tag{21}$$

Adjusting the loss function accordingly in the *neuralnet* R package results in a back-propagation neural network which emphasises the importance of the minority, rather than

having it be outweighed by the majority.

4.4.5 Chosen Methods

Given that the data considered in this thesis is rather substantial, under-sampling could be appropriate and might be rather effective as shown in Drummond et al. (2003). However, under-sampling at random could lead to an unrepresentative sample whilst under-sampling by means of clustering hardly seems feasible for such a large dataset and brings about many issues of its own, as previously discussed. Hence, under-sampling on its own will not be applied in this research. Due to its prior success, SMOTE will be considered, as well as the Multi-Minority SMOTE method proposed in this thesis in Section 4.4.2. Furthermore, two cost-sensitive methods, namely cost-sensitive boosting with decision trees as base classifier and cost-sensitive neural networks will be considered.

4.5 Performance Measures

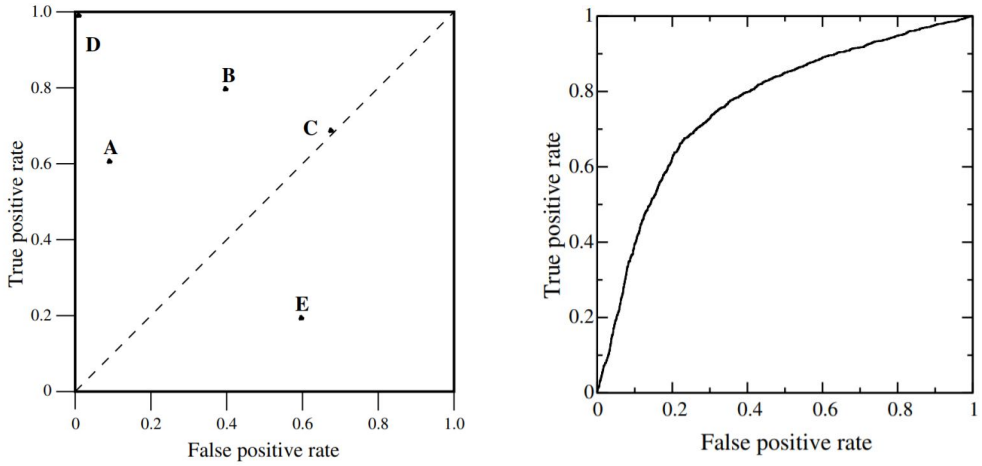
4.5.1 Confusion Table Measures

As previously mentioned, standard machine learning techniques are unable to handle imbalanced data, because they are accuracy-oriented. Accuracy measures the frequency with which observations are correctly identified, without distinguishing between the classes and therefore cannot focus on one specific class. Other measures which put more emphasis on the class of interest are sensitivity and specificity, as mentioned by Veropoulos et al. (1999). These measures are based on the two-dimensional confusion matrix as illustrated in Table 4.

Table 4: A simple confusion matrix for two classes, where TP stands for True Positive, FN stands for False Negative, FP for False Positive and TN for True Negative

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

In the table, TP is a True Positive; correctly identifying something to be true, FP is a False Positive or Type I error; incorrectly identifying something to be true, FN is a False Negative or Type II error; incorrectly identifying something to be false and lastly, TN is a True Negative; correctly identifying something to be false. Sensitivity can then be defined as the ratio of correctly defined positives out of all positives in the set: $Sensitivity = \frac{TP}{TP+FN}$ whereas specificity is defined as the ratio of correctly defined negatives: $Specificity = \frac{TN}{TN+FP}$. The confusion matrix above is two-dimensional, however, the data considered in this thesis is three-dimensional. Nevertheless, the same structure can be maintained by considering the positive class to be the class of interest, in this case the high risk class, and having the negative class be the remaining classes. Similarly, the sensitivity and specificity can be computed with respect to the other two classes, however, given the nature of the research, most value will be assigned to measures related to the high risk class.



(a) Potential results in the ROC space.

(b) An example of an ROC curve.

Figure 9: An illustration of the Receiver Operating Curve (Fawcett, 2006).

4.5.2 Area Under the ROC Curve (AUROC)

Although, in the case of skewed data, sensitivity and specificity are both better indicators of performance than accuracy, measuring them jointly leads to an even further improvement. Sensitivity can also be formulated as the true positive rate and $1 - Specificity$ is equivalent to the false positive rate. In the ideal scenario, the predictions made in a model have a true positive rate equal to one; all positive observations are identified as such, and a false positive rate equal to zero; all observations which are marked as positive are indeed positive. The ROC curve, which plots these rates against each other, as seen in Figure 9, can evaluate the predictive power of a model by considering how close it lies to the ideal point in the top left corner. The closer to this point (indicated by point D in the figure) the model's outcomes are, the better the performance of this model.

In addition, the diagonal line in Figure 9 represents the performance of randomly guessing the classes (Fawcett, 2006). This provides an extra indication of a model's performance by providing a minimum target. All points which lie below the diagonal, in this case point E, lead to a worse overall classification than random guessing would. Important to note is that there is no threshold for determining whether a particular value in the ROC plot is satisfactory or not. The ROC plots can be used to determine a model's relative performance but not absolute performance, with the exception of points which reach the ideal point.

Given that sensitivity and specificity are derived from a confusion table, one confusion table leads to one point in the ROC plot. However, when dealing with models which provide probabilities or rankings, rather than just the predicted class, multiple points can be generated by varying the threshold probability of belonging to a particular class (Fawcett, 2006). This leads to an ROC curve such as the one presented in Figure 9. A model's predictive power can then be enumerated by calculating the area under this curve, i.e. the Area Under the ROC (AUROC). The model whose AUROC value is closest to one is the superior model, according to this measure.

The prime reason for using the ROC curve in this research is that the ROC curve is insensitive to class skew (Fawcett, 2006), the previous examples were all made using binary

dependent variables, whereas this research deals with multi-class data. In order to ensure that this key property of AUROC still holds when applied to multi-class data, the approach proposed by Hand and Till (2001) is used. In their work they prove that a general one-vs-all approach of multi-class AUROC becomes sensitive to class skew, and derive a method which is not. Their measure, based on pair-wise ROC curves, is indicated by M in the following formula:

$$M = \frac{2}{c(c-1)} \sum_{i<j} \hat{A}(i,j) \tag{22}$$

where c is the number of classes and $\hat{A}(i,j)$ is the AUROC for class i versus class j.

4.5.3 Total Expected Costs

As explained in Section 4, to account for the differences in various misclassifications, cost-sensitive boosting as well as cost-sensitive neural networks are considered, which require a cost matrix. Rather than using arbitrary costs for the cost matrix, this thesis considers a cost matrix which is based on differences in expected claim size across the classes. This allows us to incorporate the fact that different errors in classification bring about different costs. In particular, the cost of not identifying an extremely risky individual is particularly abundant. In this context a false negative is deemed less severe than a false positive, i.e., classifying a high risk client as low risk is more costly than vice versa, as this mistake is also more costly for the insurance company. For this reason, if a profile is assigned which is less risky than the actual profile, the associated cost is the difference in expected claim sizes between those two classes, but if a more risky profile is assigned, then the cost is this difference halved. This structure is presented in Table 5.

The transformed costs, computed using the data and expertise of the insurance firm, are presented in Table 6[7]. It should be noted that every observation corresponds to an individual in a particular. This implies that an individual may be in one class in one year and in another the next. In standard practice this is split up even further (see Section 3) by using the exposure, which ranges from a year to a single day, to segregate observations. By aggregating by year, the risk of a potential bias in cost evaluation is decreased and moreover, the progression in risk is accounted for. Each year the individuals' age, income, social status as well as the quality of their vehicle is updated, making them a "different" person in the data. Consequently, their risk profile may also change. This is, in fact, a desirable result as the mentioned variables may be quite influential towards driving behaviour. Considering age as an example, one can imagine that an 18 year old who just obtained his/her driver's license is more prone to accidents than a 19 year old who has been driving for one year. The same applies for the elderly which is proven by the fact that, in the Netherlands, fatal accidents most frequently involve the youth and elderly (Rijkswaterstaat, 2017).

Based on these costs presented in the cost matrix, we can compute the expected total costs, even for the methods that do not incorporate cost-sensitivity.

[7]The cost matrix is scaled so as to not reveal any potentially sensitive information belonging to the insurance firm considered in this research.

Table 5: Structure of the cost matrix which is based on the difference in expected claim sizes. Assigning a less risky profile is considered more costly than vice versa, hence for the latter case the costs are halved.

		Predicted		
		Low	**Medium**	**High**
Actual	**Low**	0	(EV(Med.) - EV(Low))/2	(EV(High) - EV(Low))/2
	Medium	EV(Med.) - EV(Low)	0	(EV(High) - EV(Med.))/2
	High	EV(High) - EV(Low)	EV(High) - EV(Med.)	0

Table 6: Cost matrix according to the structure in Table 5 using anonymised expected claim sizes.

		Predicted		
		Low	**Medium**	**High**
Actual	**Low**	0	6.02	19.90
	Medium	12.04	0	13.89
	High	39.81	27.77	0

5 Results

This section discusses the results found after training the seven models by means of three fold cross-validation. As was expected, without any alterations made to the training set or objective functions, the methods failed to identify high risk policy holders. In fact, both GLM and neural networks assigned every individual to the low risk profile, with boosting being only slightly better by being able to identify a handful of medium risk clients. However, when the gradient boosting and neural networks were trained on MM SMOTE-data, the models were able to predict a much wider range of risk profile probabilities which resulted in a substantial decrease in total (expected) costs. In the case of neural networks, however, this was due to an increase in the number of medium risk profiles assigned, as it still failed to identify the high risk class. The implementation of cost-sensitivity did improve the results with regards to the original settings, but not as much as the combination with MM SMOTE did. This outcome is likely due to the nature of the objective function, which caused the model to overcompensate by hinging on the medium risk profile; the profile which, in terms of costs, is the "safest bet".

Considering the AUROC values, the total expected costs and, in particular, the ability to identify the riskiest group (High), as presented in Table 7, the model deemed most appropriate for identifying high risk policy holders in auto insurance is gradient boosting trained on MM SMOTE-data. Although this method is not superior with regards either AUROC or total expected costs, it is the only method in the top three of both measures and is able to successfully assign high risk profiles. In addition, this method has many advantages in terms of implementation as well as information extrapolation. As shown in this thesis, gradient boosting allows for non-linear relations and can approximate these relations by means of partial dependence plots. What's more, the computed variable importances provide an indication of which variables are the main contributors to an individual's risk profile.

Neural networks, although able to compute the medium risk profiles, were found to be a rather unstable method. The results as well as the convergence of this model depended strongly on the initial input parameters and furthermore, the derived insights from its output

were rather limited. The GLM, which in its simple form served mainly as a benchmark, had the benefits of simplicity and extensive feedback. It was rejected for its inability to identify individuals as anything above low risk, however, as shown in Section 5.1, the fluctuations in the probabilities it assigned did provide some indication of how risky an individual was.

The results of each method, along with their advantages and disadvantages, are discussed in more detail in the subsequent subsections.

Table 7: Overview of the average AUROC, the average total expected costs and the risk profiles which could be identified for all models considered. The best values for each measure are indicated in bold.

	AUROC	**Total Costs**	**Identified Profiles**
GLM	0.658987	6161775.82	Low
Gradient Boosting (GBM)	**0.661577**	6161703.58	Low, Medium
Neural Network (NN)	0.586816	6161775.82	Low
GBM & MM SMOTE	0.640701	5949260.81	**Low, Medium, High**
NN & MM SMOTE	0.611968	**5032532.18**	Low, Medium
Cost-Sensitive GBM	0.649831	6094418.04	**Low, Medium, High**
Cost-Sensitive NN	0.554774	5393469.16	**Low, Medium, High**

5.1 GLM

In order to determine the suitable parameters for the multinomial logistic regression, three-fold cross-validation was used on each of the three training sets, based on which the average AUROC for each combination of parameters was computed. Table 8 presents a subsample of these averages, where the best, i.e. highest, AUROC is indicated in bold (for the complete overview see Appendix B). This highest value is obtained for a regression with a shrinkage rate of $\lambda = 0.0030$ and with Ridge penalty only; $\alpha = 0$. However, as parsimonious models are preferred, rather than working with the λ value which leads to the best tuning outcome, the value is chosen such that the AUROC value differs at most one standard error from the minimum. This is in accordance with standard practice for multinomial logistic regression and in this case, the chosen parameters are $\alpha = 0$ and a shrinkage rate of $\lambda = 0.0121$, as marked in grey. In general, however, we see that especially in the case of small λ values, there is little variation in the AUROC measured, implying that there is little sensitivity with regards to the value of α chosen.

Table 8: Average AUROC for each training set as measured for various parameter values in *glmnet*. The outcome which deviates at most one standard error from the maximum is indicated in bold.

	$\alpha = 0$		$\alpha = 0.5$		$\alpha = 1$	
Lambda	**Average**	**SE**	**Average**	**SE**	**Average**	**SE**
0.0212	0.65582	0.00066	0.64267	0.00081	0.63725	0.00070
0.0160	0.65611	0.00065	0.64420	0.00085	0.63757	0.00071
0.0121	0.65631	0.00064	0.64485	0.00086	0.64030	0.00076
0.0091	0.65644	0.00063	0.64672	0.00083	0.64323	0.00078
0.0069	0.65652	0.00063	0.64823	0.00083	0.64424	0.00082
0.0052	0.65657	0.00062	0.64949	0.00084	0.64568	0.00076
0.0039	0.65659	0.00062	0.65045	0.00084	0.64730	0.00081
0.0030	0.65659	0.00061	0.65154	0.00080	0.64870	0.00081
0.0022	0.65659	0.00061	0.65244	0.00077	0.64973	0.00082
0.0017	0.65658	0.00061	0.65327	0.00076	0.65093	0.00081

In terms of insights, the multinomial logistic regression has a clear advantage. As is typical in regression, the model computes a distinct effect for each selected variable, and these coefficients can then be easily interpreted. In this sense, it is rather straightforward to determine the contribution of each variable towards the probability of belonging to a particular class. Figure 10 shows how these coefficients vary over the three classes and hence shows which coefficients are most informative for which risk profile. However, although the results displayed in Figure 10 would not immediately suggest it, the multinomial logistic regression model has difficulty in distinguishing between the three classes. In fact, when making class-based predictions, the model assigns each individual to the same class, namely the low risk class.

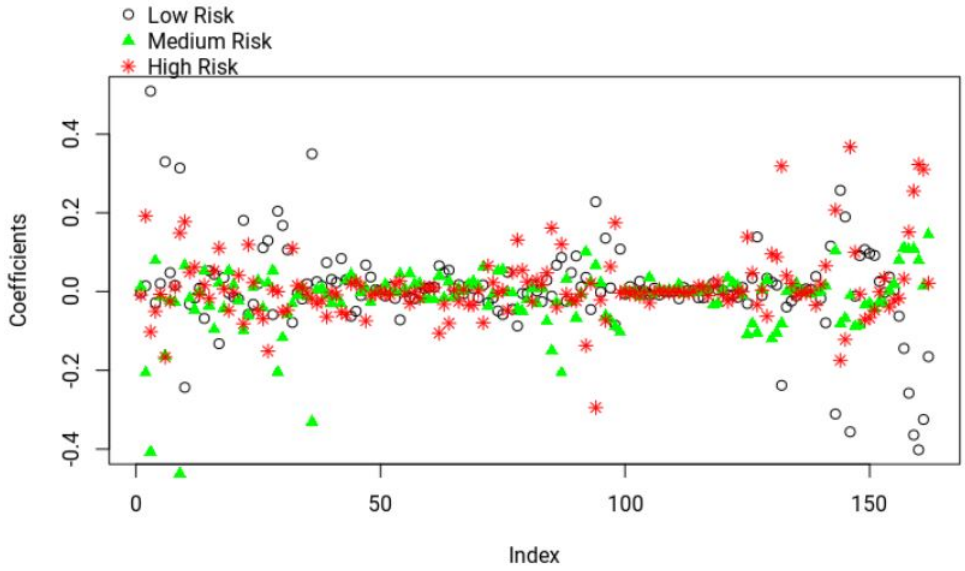


Figure 10: Overview of the GLM coefficients from the first training set, using standardised explanatory variables.

Consequently, based on class alone, we cannot determine whether the coefficients provide any information with regards to an individual's risk profile. For this purpose, we can consider the predicted probabilities rather than the predicted classes. As the low risk class is chosen each time, we know in advance that the probability of belonging to this class is the highest of the three for each individual. In fact, the probability of belonging to a class remains quite close to the sample proportion of that class, as can be seen in Table 9. Nonetheless, there are fluctuations in these probabilities, hence it seems relevant to investigate whether the relatively high probabilities in each category reflect the likelihood of belonging to that category. To do so, we can examine the proportions of the actual risk profiles in the top 10, 100 and 1000 predicted probabilities for each class. This is presented in Table 10. Note that the highest computed probability of belonging to a high risk profile is less than 5% but the results reflect to what extent differences in these low probabilities reflect riskiness.

Table 9: A sample of 20 predictions from GLM for the three risk profiles.

Low	**Medium**	**High**
0.9040	0.0666	0.0293
0.9025	0.0690	0.0285
0.8282	0.1333	0.0385
0.9402	0.0472	0.0126
0.8932	0.0802	0.0266
0.9654	0.0279	0.0067
0.8937	0.0738	0.0325
0.9263	0.0503	0.0234
0.9563	0.0322	0.0115
0.9457	0.0420	0.0123
0.8927	0.0815	0.0258
0.9221	0.0599	0.0180
0.9461	0.0390	0.0149
0.9285	0.0530	0.0186
0.9192	0.0610	0.0198
0.9485	0.0412	0.0103
0.9048	0.0746	0.0206
0.9389	0.0461	0.0150
0.9625	0.0295	0.0080
0.9825	0.0138	0.0037

As the table shows, the fluctuations in probabilities do not truly capture the differences in risk profiles, however, there is some information hidden in these changes. The model is particularly good at distinguishing the low risk individuals as such. In addition, of the ten individuals with the highest probability of being high risk, on average, two are indeed high risk. In the top 100 this is 18% and in the top 1000 this is 11.33%. These percentages are not particularly high in themselves, but is a significant increase with regards to the 1.9% sample proportion of high risk drivers.

Table 10: Average proportions of risk profiles observed in the top 10, top 100 and top 1000 probabilities of belonging to each class, as computed in GLM.

Top 10	**Low**	**Medium**	**High**
Pr(Low)	100%	0%	0%
Pr(Medium)	56.67%	26.67%	16.67%
Pr(High)	63.33%	16.67%	20%
Top 100	**Low**	**Medium**	**High**
Pr(Low)	98.67%	1.33%	0%
Pr(Medium)	64.33%	20.33%	15.33%
Pr(High)	67%	15%	18%
Top 1000	**Low**	**Medium**	**High**
Pr(Low)	98.87%	1%	0.13%
Pr(Medium)	71.57%	18.33%	10.1%
Pr(High)	73.8%	15.07%	11.33%

Table 11: Confusion tables for each fold resulting from the trained multinomial logit models.

	Fold 1			**Fold 2**			**Fold 3**		
	Low	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Low	426541	0	0	426663	0	0	426500	0	0
Medium	25778	0	0	26104	0	0	25910	0	0
High	8944	0	0	8495	0	0	8852	0	0

5.2 Gradient Boosting

Rather than measuring coefficients, as is done in GLM, the gradient boosting method evaluates the importance of variables by means of their relative influence on the outcomes, as explained in Section 4.2.2. The average relative influence of the variables across the three training sets is presented in Table 12. The top three variables, which represent characteristics of the individual drivers, are of particular importance, whilst various car variables are of secondary importance. This implies that several individual characteristics are most effective in assigning risk profiles. These influences were found in a gradient boosting model using an interaction depth of eight, one thousand trees and a shrinkage of 0.001, based on the tuning results shown in the Appendix in Table 34.

Given that transparency is key in the field of insurance, the depth of insights provided by the methods is important. The influences shown above may not have the added benefits of having statistical properties such as significance (although determining significance is also not straightforward in the lasso, and hence not in the elastic net technique (Lockhart et al., 2014)), a definite advantage is the non-linearity of these effects. Figure 11 shows the impact of one particular variable on the probability of belonging to each class respectively, taking the effect of one value of this variable as reference point. By doing so we can clearly see that this variable has a marked impact on the probability of belonging to the classes, and that it makes a clear distinction between the medium and high risk profiles.

Considering the predictive power of the model, Table 13 shows that, in contrast to GLM, gradient boosting can also identify the medium risk profile in a handful of cases. However, the probability of being high risk remains rather restricted. The improved ability

Table 12: Average relative influence of the variables across the three training sets, as found in gradient boosting.

Variable	**Average Relative Influence**
Individual B	42.273
Individual E	38.690
Individual F	7.559
Car F	5.567
Car I	2.179
Car E	1.463
Car H	0.774
Car A	0.591
Car B	0.301
Car G	0.211
Individual C	0.129
Social F	0.068
Car D	0.052
Social C	0.051
Car J	0.047
Social B	0.027
Car C	0.008
Social G	0.004
Social E	0.005
Social A	0.002
Individual A	0.000
Social D	0.000
Individual D	0.000

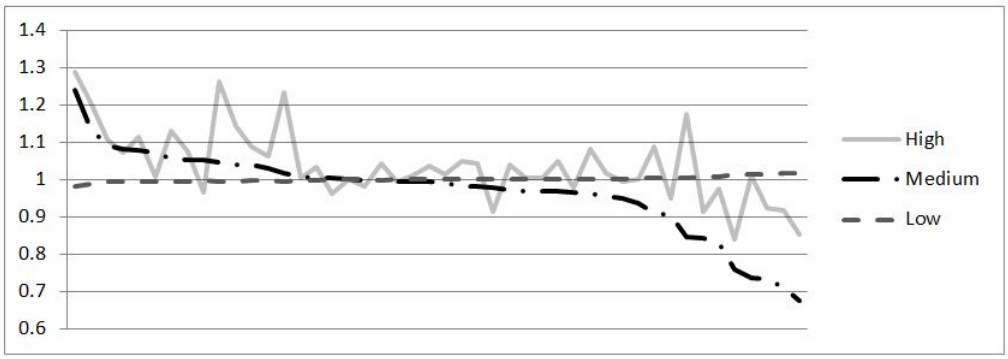


Figure 11: Overview of the GLM coefficients from the first training set, using standardised explanatory variables.

Table 13: Confusion tables for each fold resulting from the trained gradient boosting.

	Fold 1			**Fold 2**			**Fold 3**		
	Low	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Low	426839	2	0	426661	2	0	426496	4	0
Medium	25774	4	0	26101	3	0	25907	3	0
High	8944	0	0	8491	4	0	8850	2	0

of predicting medium classes is also reflected in the larger proportions for this profile in Table 14 as compared to those presented in Table 10. This improvement does, however, seem to be at the cost of the accuracy of distinguishing the group most at risk.

Table 14: Average proportions of risk profiles observed in the top 10, top 100 and top 1000 probabilities of belonging to each class, as computed in GBM.

Top 10	**Low**	**Medium**	**High**
Pr(Low)	100%	0%	0%
Pr(Medium)	33.33%	43.33%	23.33%
Pr(High)	70%	16.67%	13.33%
Top 100	**Low**	**Medium**	**High**
Pr(Low)	99.33%	0.66%	0%
Pr(Medium)	58%	29%	13%
Pr(High)	65%	21%	14%
Top 1000	**Low**	**Medium**	**High**
Pr(Low)	98.87%	0.9%	0.23%
Pr(Medium)	68.67%	21.53%	9.8%
Pr(High)	70.7%	17.4%	11.9%

5.3 Neural Networks

After having tuned the neural networks for various amounts of neurons in the hidden layer, it was found that using 60 neurons was most successful, as seen in Table 38. Neural networks, being rather intensive in computation power, are also quite limited in their interpretability. However, using the method described in Section 4.3, we can derive the extent to which a variable contributes positively towards the AUROC. Table 15 shows that, as seen in the previous two methods, the car and individual variables are most influential and we see a similar selection of these variables at the top of the list.

Table 15: The average sensitivities of the trained neural networks, as described in Section 4.3

Variable	**Sensitivity**
Individual Variable E	0.06075
Individual Variable B	0.03971
Car Variable F	0.01221
Individual Variable F	0.00666
Car Variable B	0.00577
Car Variable J	0.00430
Car Variable C	0.00400
Social Variable G	0.00277
Car Variable D	0.00247
Car Variable E	0.00232
Car Variable I	0.00187
Car Variable A	0.00104
Social Variable C	0.00077
Social Variable B	0.00045
Social Variable E	0.00041
Individual Variable C	0.00038
Individual Variable D	0.00037
Car Variable G	0.00013
Social Variable A	0.00008
Individual Variable A	0.00006
Social Variable D	0.00002
Car Variable H	0
Social Variable F	0

Despite the complexity of the method, the confusion tables in Table 16 show that the predictive power is rather disappointing. In all three folds only the low class is predicted, as was the case for the GLM model. Additionally, the fluctuations of these probabilities do not reflect the inherent riskiness of consumers well, much poorer than those of the GLM probabilities. It is clear here that the one-layer neural network is not able to deal with the class skew.

Table 16: Confusion tables for each fold resulting from the trained neural networks.

	Fold 1			**Fold 2**			**Fold 3**		
	Low	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Low	426541	0	0	426663	0	0	426500	0	0
Medium	25778	0	0	26104	0	0	25910	0	0
High	8944	0	0	8495	0	0	8852	0	0

Table 17: Average proportions of risk profiles observed in the top probabilities of belonging to each class, as computed in the neural network.

Top 10	**Low**	**Medium**	**High**
Pr(Low)	98.15%	0.62%	1.23%
Pr(Medium)	84.45%	8.44%	7.12%
Pr(High)	84.45%	8.44%	7.12%

5.4 MM SMOTE

As discussed in the previous sections, one of the main difficulties of the data considered in this thesis, is the skewness thereof. Due to this imbalance, most methods are not able to distinguish between the classes, focusing only on the low risk profile. To improve this balance, the newly presented MM SMOTE is applied (see Section 4.4.2). This method takes a randomised subsample of the low risk group and generates additional synthesised observations of the medium and high risk groups. This leads to the following proportions of classes:

Table 18: The proportions of the three risk profiles before and after applying MM SMOTE.

	Low Risk	**Medium Risk**	**High Risk**
Original	92.5%	5.6%	1.9%
After MM SMOTE	52.7%	31.4%	15.9%

The observations are still not distributed evenly amongst the classes, however this is a conscious choice. The aim is to see whether a more even distribution of classes, whilst maintaining the ordering of the frequencies (low risk is most frequent and high risk is least frequent), would lead to an improvement in the test set. The reason for keeping the ordering intact, is that changing the frequencies from 92.5%, 5.6% and 1.9% to one with an equal distribution would require drastic alterations of the data.

5.4.1 MM SMOTE & Gradient Boosting

Whereas gradient boosting on its own failed to identify any individual as high risk, as expected due to the model's assumption of class balance, when used in combination with MM SMOTE, it is now able to do so. Table 19 shows that the predictions on the test set are much more spread out than before, and that the model is better able to distinguish between the classes. What is noticeable is that although the model is trained on a balanced dataset, it's predictions in the imbalanced test sets reflect the true proportions. In other words,

despite being trained on a balanced dataset, it does not enforce this equal distribution in its predictions, suggesting that it does in fact recognise the properties of the various risk profiles. Nevertheless, the high risk profile is assigned incorrectly more often than it is assigned correctly, as is reflected in Table 20.

Table 19: Confusion tables for each fold resulting from gradient boosting machines trained on MM SMOTE-data.

	Fold 1			**Fold 2**			**Fold 3**		
	Low	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Low	410733	15679	129	412345	14147	171	410228	16065	207
Medium	23278	2484	16	23732	2350	22	23351	2521	38
High	7881	1051	12	7594	880	21	7798	1037	17

Table 20: Average proportions of risk profiles observed in the top 10, top 100 and top 1000 probabilities of belonging to each class, as computed in GBM after MM SMOTE.

Top 10	**Low**	**Medium**	**High**
Pr(Low)	100%	0%	0%
Pr(Medium)	73.33%	13.33%	13.33%
Pr(High)	83.33%	16.67%	0%
Top 100	**Low**	**Medium**	**High**
Pr(Low)	98%	1.33%	0.67%
Pr(Medium)	80.67%	11.67%	7.67%
Pr(High)	82.67%	13%	7.67%
Top 1000	**Low**	**Medium**	**High**
Pr(Low)	98.73%	1%	0.27%
Pr(Medium)	81.6%	13.33%	5.67%
Pr(High)	78.1%	13.2%	8.7%

Table 21 shows the average relative importance of the variables found by applying gradient boosting on the MM SMOTE-data training sets. The parameters used for this purpose were an interaction depth of eight, eight hundred trees and a shrinkage of 0.01. Variable *Individual B* remains at the top of the list, but with respect to the ranking found in Table 12, this ranking appears to put more emphasis on the variables related to the cars. In addition, it is particularly evident that the post-code based variables, i.e. *Social* variables do not have a strong impact on the segregation of classes.

Table 21: Average relative influence of the variables across the three MM SMOTE-data training sets, as found in gradient boosting.

Variable	**Average Relative Influence**
Individual B	22.838
Car E	19.747
Car B	16.335
Car D	7.780
Car F	6.172
Car C	4.461
Individual D	3.987
Individual E	2.669
Car H	2.882
Individual C	2.433
Car G	2.489
Car J	1.900
Car I	1.847
Individual F	1.544
Car A	1.412
Social F	0.488
Social A	0.421
Social G	0.334
Social B	0.123
Social E	0.093
Social C	0.054
Individual A	0.002
Social D	0.000

5.4.2 MM SMOTE & Neural Networks

Having tuned the model for various number of neurons, again 60 neurons in the hidden layer turns out to be the most successful 40. Based on this model, the model was trained on the MM SMOTE-datasets and predictions were made on the non-SMOTE data. Table 22 shows that by using MM SMOTE, the model is now able to predict a substantial number of medium risk profiles, as compared to only low risk when not using MM SMOTE. Nonetheless, the model is still incapable of predicting any high risk individuals and is therefore deemed inappropriate for the purpose of predicting high risk individuals.

Table 22: Confusion tables for each fold resulting from neural networks trained on MM SMOTE-data.

	Fold 1			**Fold 2**			**Fold 3**		
	Low	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Low	339460	87081	0	349987	76553	0	280899	145641	0
Medium	17320	8458	0	21124	4654	0	16850	8928	0
High	5448	3496	0	7374	1570	0	5935	3009	0

5.5 Cost-Sensitive Boosting

Whereas MM SMOTE handles class skew by adjusting the training set, the cost-sensitive methods aim to do so by incorporating the cost matrix in their objective functions. As a result, the methods become quite computationally intensive and due to limitations of the programming environment, only stubs, i.e. one-layer decision trees, could be considered for cost-sensitive boosting. These circumstances also made tuning slightly problematic, however, it was found that one hundred trees worked best as shown in the appendix in Table 42.

Once again the variable influence could be computed, this time using the same method as that which is used for the neural networks. This is because the *gbm* package is not used for this method but instead, MATLAB code has been converted to R code, not including the computations for the influence or partial dependence plots. The latter was excluded as it was outside of the scope of this research and it did not fit into the already ambitious time line for this thesis. Subsequently, the degree of insights provided by this method is more restricted than the stand-alone gradient boosting machine, or that combined with MM SMOTE.

Table 23: The average sensitivities, as described in Section 4.3, of the variables used in cost-sensitive boosting.

Variable	**Sensitivity**
Individual Variable E	0.01047
Car Variable A	0.00230
Individual Variable B	0.00203
Social Variable E	0.00119
Individual Variable F	0.00099
Car Variable E	0.00057
Social Variable D	0.00054
Car Variable G	0.00035
Car Variable J	0.00033
Social Variable C	0.00025
Car Variable F	0.00017
Car Variable D	0
Car Variable I	0
Car Variable B	0
Car Variable C	0
Individual Variable D	0
Social Variable G	0
Social Variable F	0
Individual Variable C	0
Social Variable A	0
Social Variable B	0
Individual Variable A	0
Car Variable H	0

In terms of performance, cost-sensitive boosting can predict the medium class quite abundantly, and although it does manage to predict some high class individuals successfully, this

is a rare occurrence.

Table 24: Confusion tables for each fold resulting from the trained cost-sensitive boosting model.

	Fold 1			**Fold 2**			**Fold 3**		
	Low	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Low	422136	4527	0	421760	4781	0	420993	5547	0
Medium	25035	1068	1	24720	1056	2	25010	769	1
High	8041	452	2	8455	488	2	8322	621	1

Using the deciles presented in Table 25, one can see that the low risk profile still dominates the predictions. However, over 10% of the top 10, 100 and 1000 probabilities of belonging to the high risk class correctly reflect the riskiness of those drivers. This implies that the fluctuations in probabilities do reveal some truths about the risk profiles of the individuals considered in this research.

Table 25: Average proportions of risk profiles observed in the top 10, top 100 and top 1000 probabilities of belonging to each class, as computed in cost-sensitive gradient boosting.

Top 10	**Low**	**Medium**	**High**
Pr(Low)	100%	0%	0%
Pr(Medium)	66.67%	26.67%	6.67%
Pr(High)	70%	16.67%	13.33%
Top 100	**Low**	**Medium**	**High**
Pr(Low)	99.33%	0.66%	0%
Pr(Medium)	68.67%	22.67%	8.67%
Pr(High)	75.67%	12.33%	12%
Top 1000	**Low**	**Medium**	**High**
Pr(Low)	98.8%	1.67%	0.13%
Pr(Medium)	71.73%	19.73%	9.93%
Pr(High)	77.26%	12.72%	10.02 %

5.6 Cost-Sensitive Neural Networks

The final method presented in this thesis is the cost-sensitive neural network. Thusfar, the high expectations for the neural network methods have not been met as they have so far been unable to detect any members of the high risk class in the data. This cost-sensitive version, made by adjusting the *neuralnet* function according to the work by Kukar et al. (1998), is very computationally intensive (requiring weeks to tune) but does allow a neural network to identify high risk profiles. The ability to do so is quite dependent on the fold, as in the first fold no high risk profiles are assigned, but in the third fold 16 are assigned.

It appears as though the method overcompensates for the different cost structure by focussing on the medium risk profile, which may be considered a safe option in terms of minimising costs. Another issue is illustrated by the deciles presented in Table 27. These show that the largest assigned probabilities for high risk in this auto insurance data poorly reflect the inherent riskiness of the drivers assigned these probabilities. In general, it can be concluded that the backpropagation neural network is not well suited to identify the high

risk class in this particular car insurance policy. What's more, this poor performance is exacerbated by the restricted insights it provides.

Table 26: Confusion tables for each fold resulting from the trained cost-sensitive neural networks.

	Fold 1			**Fold 2**			**Fold 3**		
	Low	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Low	366719	59819	3	306086	120545	32	357994	67808	698
Medium	20830	4947	1	16421	9682	1	19651	6195	64
High	7032	1912	0	5138	3356	1	6504	2332	16

Table 27: Average proportions of risk profiles observed in the top 10, top 100 and top 1000 probabilities of belonging to each class, as computed in the cost-sensitive neural network.

Top 10	**Low**	**Medium**	**High**
Pr(Low)	93.33%	6.67%	0%
Pr(Medium)	93.33%	6.67%	0%
Pr(High)	93.33%	6.67%	0%
Top 100	**Low**	**Medium**	**High**
Pr(Low)	96%	4%	0%
Pr(Medium)	89.33%	8.33%	2.33%
Pr(High)	89%	9%	2%
Top 1000	**Low**	**Medium**	**High**
Pr(Low)	97.03%	2.60%	0.37%
Pr(Medium)	87.93%	8.6%	3.47%
Pr(High)	90.43%	6.83%	2.73%

6 Conclusions and Further Research

This thesis presents a new approach to investigating claims in the field of auto insurance. This is done by focusing on the level of risk per individual, rather than the frequency or severity of claims, as is common in this sector. Instead, these measures are used to assign individuals to one of three risk profiles, namely low, medium and high risk, where the latter, consisting of merely 1.9% of the sample, is of key importance. In addition to converting the standard approach to a classification problem, this research has investigated the potential benefits of applying machine learning techniques as compared to a generalised linear model, which is currently used. Furthermore, given the necessity to defend policy decisions in insurance and the fact that machine learning techniques are often criticised for being a "black box", each method is also evaluated in terms of their ability to provide insights. The main aim of this thesis is therefore to determine whether recent data-mining techniques can trump standard practice in the prediction policy holder risk in auto insurance.

A straightforward GLM model serves as benchmark, whereas the selected data-driven techniques are gradient boosting and a backpropagation neural network. In order to deal with the severe class imbalance, as shown in Section 3, the latter methods are also run in combination with MM SMOTE, which balances the distribution of classes by generating synthetic copies of the medium and high risk observations and including only a subset of the low risk observations. MM SMOTE was introduced in this thesis as an alteration of the original SMOTE algorithm, which focuses only on binary cases, so as to oversample the minority classes proportionally. In addition, to account for the difference in severity of different types of mistakes, cost-sensitive adaptations of boosting and neural networks are applied.

The GLM model, although advantageous in terms of the statistical insights it presents, failed to identify any class other than the low risk profile, proving its inability to handle the class imbalance. Consequently, boosting and neural networks are found to be better able to differentiate between the risk profiles of policy holders, in particular when they were combined with the Multi-Minority SMOTE algorithm presented in this thesis, or with a cost-sensitive objective function. The methods were evaluated based on four aspects; the average AUROC over the three folds, the average total expected costs, the range of risk profiles identified, as well as their ability to provide insights regarding their decision making process. In terms of AUROC, the stand-alone version of gradient boosting is most successful, achieving an average value of 0.661577, whereas neural networks trained on MM SMOTE-data minimised the total expected costs at 5,032,532.18. However, neither of these methods managed to predict the high risk profiles. Given that the focus of this research lies with the identification of high risk clients and the need for transparency in the field of insurance, gradient boosting combined with MM SMOTE is deemed most appropriate for the problem at hand.

The results of this thesis show that substantial improvements in the identification of high risk individuals can be made by resorting to data-mining techniques. However, it should be noted that the purity of the classifications remains quite troubling. Although gradient boosting in combination with MM SMOTE is deemed most successful at assigning risk profiles to policy holders, many are still misclassified. Hence, the results do not warrant a change in pricing policy, but instead should be used as an indication that some individuals may be more risky and hence should be considered more carefully.

Lastly, in addition to presenting a new approach in the car insurance sector, this thesis presents new empirical results in a developing field of research. The application of cost-

sensitive, multi-class data mining techniques on imbalanced data has thus far been quite limited. Particularly on data of the scale such as that considered in this thesis. The results of this research show that class imbalance can present a serious problem to classification techniques, and provides empirical evidence that this problem can be (somewhat) alleviated by the incorporation of techniques such as MM SMOTE and cost-sensitivity.

6.1 Limitations

One of the main advantages of this research was the availability of such substantial data, which included the entire portfolio of one particular car insurance over several years. Nonetheless, the value of these data lead to various drawbacks. Given that the data considered in this thesis were confidential, several measures had to be taken within the programming environment in order to ensure proper security. One of those measures was to run all models in a secure R server provided by the company. The main disadvantage of this server was that the version thereof was quite outdated, namely version 3.0.1. dating from 2013. As a result, various R packages and functionalities could not be applied as they were not compatible with this version. In addition, a secure VPN connection was needed to access the R server, but the connection time thereof was limited. Consequently, there was a cap on the running time as R would quit its process once disconnected. Although ample time and effort were put into finding compatible alternatives or alterations of the methods required for this research, some functions were simply not available. In addition, due to the limitations on the running time, the potential of the computationally intensive cost-sensitive methods could not be fully investigated. Nevertheless, by splitting up various procedures, in particular for the tuning process, several issues could be circumvented.

As mentioned in the methodology section, some compromises had to be taken for the cost-sensitive methods. Firstly, the cost-sensitive boosting method was only able to handle stumps, as the option to convert code to C (which would allow for faster computations) was blocked for security reasons. Computing deep trees on such substantial data was not feasible and sensitive to crashes in R. For similar reasons, neural networks with only one hidden layer were considered. Despite their computational intensity, it was possible to tune various scenarios for these methods, however, in less rigid circumstances the tuning process would have been expanded to cover a wider range of parameter values. It should also be mentioned that, given that (cost-sensitive) neural networks have difficulties converging, running the models on subsets of the data for the purpose of parameter tuning often led to non-optimal outcomes where each individual was assigned the same set of probabilities.

6.2 Further Research

As mentioned above, due to restrictions on the programming environment, some compromises needed to be made. Hence, the primary suggestions for further research would be to extend on the methods presented in this thesis. This could be done by considering cost-sensitive boosting with deeper trees as well as neural networks with a multitude of hidden layers, i.e. deep learning. Moreover, due to the lack of consensus regarding the application of neural networks along with the required data preparation, the sensitivity of the results regarding to these processes could be investigated.

This research started with the simple question to investigate extremes within car insurance, whenceforth the risk profiles, data preparation, methodology and cost-sensitivity followed. Although these ideas were discussed with and approved by experts in actuarial

science, alterations could be made. It may, for example, be of interest to consider a different number of risk profiles or to change the guidelines used to assign individuals to their respective profiles. As a result, the nature of the cost matrix would also change.

In order to deal with the severe imbalance in the data, this thesis used a newly introduced multi-class version of SMOTE, dubbed MM SMOTE, as well as cost-sensitivity. However, recently an alternative to gradient boosting, called Delta Boosting, has been introduced which in itself is more robust to class skew (Lee et al., 2015). This method has shown potential in the field of insurance and hence it would be interesting to compare how this method fares in identifying risk.

Lastly, an extension of this research could be to enrich the data by means of telematics; a relatively new development in which a black box records the driving behaviour of the car (Baecke and Bocca, 2017). With telematics we can expand the rather static information we have on individuals with dynamic measures such as the frequency with which one drives, how often one brakes, where one drives, etc. Consequently, it is much easier to gain an accurate depiction of the risk someone takes on the road and hence assign appropriate risk profiles. Seeing as the dimensions of the data would increase substantially, it would be critical to combine the methods presented in this thesis with some form of variable selection or dimension reduction such as principal components analysis.

6.3 Practical Implications

This thesis has presented an original approach to dealing with auto insurance data and could be a starting point for new fields of research. Whereas conventional auto insurance practices focus on predicting either claim frequency or claim severity, this thesis has put forth the concept of risk profiles, and focuses on the identification of extremely risky policy holders. Besides gaining insights regarding the characteristics of risky clients, there are various other possible applications in the field of insurance. A straightforward application is that of pricing. Building on the notion of expected costs discussed in Section 4.5.3, one could assign distinct costs to the variables, depending on their contribution to the overall likeliness of belonging to the medium or high risk class. An individual's premium could then be, to some extent, based upon the combination of their personal attributes and the designated costs of these attributes. This is similar to the current approach in the industry, but also incorporates the discontinuous nature of the risk segments.

Moreover, the identification of high risky individuals can also be useful whilst assessing potential new clients. Individuals assigned to a high risk profile may need to provide additional evidence of their driving history or may need to meet additional requirements in order to be accepted by the insurer. For current clients, awareness of their risk profiles may allow for a higher degree of efficiency in claim inspections. Not every claim is inspected for fraud, but instead, a selection of claims are inspected at random or based on a client's claim history. Given that the risk profiles are (partially) derived from this claim history, a more sophisticated approach may be to perform these additional checks more frequently amongst the high risk clients.

References

Abbring, J. H., Chiappori, P.-A., and Zavadil, T. (2008). Better safe than sorry? ex ante and ex post moral hazard in dynamic insurance data.

Anderson, D., Feldblum, S., Modlin, C., Schirmacher, D., Schirmacher, E., and Thandi, N. (2004). A practitioner's guide to generalized linear models. *Casualty Actuarial Society Discussion Paper Program*, pages 1–116.

Appel, R., Burgos-Artizzu, X., and Perona, P. (2016). Improved multi-class cost-sensitive boosting via estimation of the minimum-risk class. *arXiv preprint arXiv:1607.03547*.

Arrow, K. (1963). Uncertainty and the welfare economics of medical care, 53 am. *Econ. Rev*, 941:947–965.

Baecke, P. and Bocca, L. (2017). The value of vehicle telematics data in insurance risk selection processes. *Decision Support Systems*, 98:69 – 79.

Beijbom, O., Saberian, M., Kriegman, D., and Vasconcelos, N. (2014). Guess-averse loss functions for cost-sensitive multiclass boosting. In *International Conference on Machine Learning*, pages 586–594.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

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

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984). *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis.

Brockman, M. J. and Wright, T. (1992). Statistical motor rating: making effective use of your data. *Journal of the Institute of Actuaries*, 119(03):457–543.

Chapados, N., Bengio, Y., Vincent, P., Ghosn, J., Dugas, C., Takeuchi, I., and Meng, L. (2002). Estimating car insurance premia: A case study in high-dimensional data inference. In *Advances in Neural Information Processing Systems*, pages 1369–1376.

Chawla, N., Lazarevic, A., Hall, L., and Bowyer, K. (2003). Smoteboost: Improving prediction of the minority class in boosting. *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Chiappori, P. and Salanie, B. (2000). Testing for asymmetric information in insurance markets. *Journal of Political Economy*, 108(1):56–78.

Cohen, A. (2005). Asymmetric information and learning: Evidence from the automobile insurance market. *The Review of Economics and Statistics*, 87(2):197–207.

Cost, S. and Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine learning*, 10(1):57–78.

Cummins, J. D. and Tennyson, S. (1996). Moral hazard in insurance claiming: Evidence from automobile insurance. *Journal of Risk and Uncertainty*, 12(1):29–50.

David, M. (2015). Auto insurance premium calculation using generalized linear models. *Procedia Economics and Finance*, 20:147 – 156. Globalization and Higher Education in Economics and Business Administration - GEBA 2013.

Dawson, C. W. and Wilby, R. (1998). An artificial neural network approach to rainfall-runoff modelling. *Hydrological Sciences Journal*, 43(1):47–66.

De Jong, P., Heller, G. Z., et al. (2008). *Generalized linear models for insurance data*, volume 10. Cambridge University Press Cambridge.

Desyllas, P. and Sako, M. (2013). Profiting from business model innovation: Evidence from pay-as-you-drive auto insurance. *Research Policy*, 42(1):101 – 116.

Drummond, C., Holte, R. C., et al. (2003). C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11. Citeseer Washington DC.

Dutang, C. (2016). *Insurance datasets*. R package version 1.0-5.

Edgar Acuna, E. and Rodriguez, C. (2004). The treatment of missing values and its effect in the classifier accuracy. In *Proceedings of the Meeting of the International Federation of Classification Societies (IFCS)*, pages 639–47.

Fan, W., Stolfo, S. J., Zhang, J., and Chan, P. K. (1999). Adacost: misclassification cost-sensitive boosting. In *Icml*, volume 99, pages 97–105.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861 – 874. ROC Analysis in Pattern Recognition.

Francis, L. (2001). Neural networks demystified. In *Casualty Actuarial Society Forum*, pages 253–320.

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1.

Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Guelman, L. (2012). Gradient boosting trees for auto insurance loss cost modeling and prediction. *Expert Syst. Appl.*, 39(3):3659–3667.

Han, H., Wang, W.-Y., and Mao, B.-H. (2005a). Borderline-smote: a new over-sampling method in imbalanced data sets learning. *Advances in intelligent computing*, pages 878–887.

Han, H., Wang, W.-Y., and Mao, B.-H. (2005b). *Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning*, pages 878–887. Springer Berlin Heidelberg, Berlin, Heidelberg.

Hand, D. J. and Till, R. J. (2001). A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45(2):171–186.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Hsu, C.-W. and Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425.

Kang, H. (2013). The prevention and handling of the missing data. *Korean journal of anesthesiology*, 64(5):402–406.

Kowarik, A. and Templ, M. (2016). Imputation with the R package VIM. *Journal of Statistical Software*, 74(7):1–16.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957.

Kuhn, M., with contributions from Wing, J., Weston, S., Williams, A., Keefer, C., Engelhardt, A., Cooper, T., Mayer, Z., Kenkel, B., the R Core Team, Benesty, M., Lescarbeau, R., Ziem, A., Scrucca, L., Tang, Y., Candan, C., and Hunt., T. (2017). *caret: Classification and Regression Training*. R package version 6.0-76.

Kukar, M., Kononenko, I., et al. (1998). Cost-sensitive learning with neural networks. In *ECAI*, pages 445–449.

Lee, S., Lin, S., and Antonio, K. (2015). Delta boosting machine and its application in actuarial modeling.

Liaw, A., Wiener, M., et al. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.

Lockhart, R., Taylor, J., Tibshirani, R. J., and Tibshirani, R. (2014). A significance test for the lasso. *Annals of statistics*, 42(2):413.

Lusa, L. et al. (2013). Smote for high-dimensional class-imbalanced data. *BMC bioinformatics*, 14(1):106.

Mani, I. and Zhang, I. (2003). knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, volume 126.

Milborrow, S. (2018). *rpart.plot: Plot 'rpart' Models: An Enhanced Version of 'plot.rpart'*. R package version 3.0.5.

Mingers, J. (1989). An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3(4):319–342.

Mosley Jr, R. C. (2004). Estimating claim settlement values using glm. *Casual Actuary Society "Applying and Evaluating Generalized Linear Models*, pages 291–314.

Natekin, A. and Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7.

Ohlsson, E. and Johansson, B. (2010). *Non-life insurance pricing with generalized linear models*, volume 21. Springer.

Qazi, N. and Raza, K. (2012). Effect of feature selection, smote and under sampling on class imbalance classification. In *Computer Modelling and Simulation (UKSim), 2012 UKSim 14th International Conference on*, pages 145–150. IEEE.

Rahman, M. M. and Davis, D. (2013). Addressing the class imbalance problem in medical datasets. *International Journal of Machine Learning and Computing*, 3(2):224.

Reuters (2016). Google looks to partner with insurance companies in France. `http://fortune.com/2016/09/13/google-france-insurance-partners/`. Accessed: 2017-07-31.

Ridgeway, G. and with contributions from others (2017). *gbm: Generalized Boosted Regression Models*. R package version 2.1.3.

Rijkswaterstaat, C. (2017). Omgekomen autorijders vaak jong of oud. `https://www.cbs.nl/nl-nl/nieuws/2017/18/omgekomen-autorijders-vaak-jong-of-oud`. Accessed: 2018-11-05.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252.

S. Sarle, W. (2002). Neural network faq, part 2 of 7: Learning, periodic posting to the usenet newsgroup comp.ai.neural-nets.

Schapire, R. E. (1990). The strength of weak learnability. *Mach. Learn.*, 5(2):197–227.

Shanker, M., Hu, M. Y., and Hung, M. S. (1996). Effect of data standardization on neural network training. *Omega*, 24(4):385–397.

Simon, N., Friedman, J., Hastie, T., and Tibshirani, R. (2011). Regularization paths for cox's proportional hazards model via coordinate descent. *Journal of Statistical Software*, 39(5):1–13.

Starkweather, J. and Moske, A. K. (2011). Multinomial logistic regression. *Consulted page at September 10th: http://www. unt. edu/rss/class/Jon/Benchmarks/MLR_JDS_Aug2011. pdf*, 29:2825–2830.

Sun, Y., Kamel, M. S., Wong, A. K., and Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378.

Tahir, M. A., Kittler, J., Mikolajczyk, K., and Yan, F. (2009). *A Multiple Expert Approach to the Class Imbalance Problem Using Inverse Random under Sampling*, pages 82–91. Springer Berlin Heidelberg, Berlin, Heidelberg.

Therneau, T. and Atkinson, B. (2018). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-13.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. ISBN 0-387-95457-0.

Veropoulos, K., Campbell, C., and Cristianini, N. (1999). Controlling the sensitivity of support vector machines. In *Proceedings of the International Joint Conference on AI*, pages 55–60.

Wang, J., Xu, M., Wang, H., and Zhang, J. (2006). Classification of imbalanced data by using the smote algorithm and locally linear embedding. In *Signal Processing, 2006 8th International Conference on*, volume 3. IEEE.

Wasserman, P. D. and Schwartz, T. (1988). Neural networks. ii. what are they and why is everybody so interested in them now? *IEEE Expert*, 3(1):10–15.

Yen, S.-J. and Lee, Y.-S. (2009). Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, 36(3, Part 1):5718 – 5727.

Zheng, J. (2010). Cost-sensitive boosting neural networks for software defect prediction. *Expert Systems with Applications*, 37(6):4537–4543.

Zhou, Z.-H. and Liu, X.-Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):63–77.

Zhu, J. and Hastie, T. (2004). Classification of gene microarrays by penalized logistic regression. *Biostatistics*, 5(3):427–443.

Zhu, J., Zou, H., Rosset, S., Hastie, T., et al. (2009). Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

A Multi-Minority SMOTE Code

```
SMOTE.M <- function(form, data,
                    perc.over=200,k=5,
                    perc.under=200,
                    equal.over=FALSE,
                    learner=NULL,
                    numMin=1,
                    ...) {

  tgt <- which(names(data) == as.character(form[[2]]))
  minClasses = NULL
  for(i in 1:numMin) {
    l <- levels(data[,tgt])[!levels(data[,tgt]) %in% minClasses]
    t <- table(data[data[,tgt] %in% l,tgt])
    t <- t[t!= 0]
    mC <- l[which.min(t)]
    minClasses <- c(minClasses, mC)
  }

  if(equal.over == FALSE) {
    totalMin <- length(which(data[,tgt] %in% minClasses))
    perc.over.i <- sapply(minClasses, function(m) {
      perc <- 1 - (length(which(data[,tgt] == m))/totalMin)
    })
    totalPerc <- sum(perc.over.i)
    perc.over.m <- sapply(minClasses, function(m) {
      per <- round(numMin * perc.over * perc.over.i[m]/totalPerc)
      per
    })
    names(perc.over.m) <- names(perc.over.i)
  } else {
    perc.over.m <- sapply(minClasses, function(m) {
      perc <- perc.over
      perc})
  }

  newExs <- NULL
  minExsM <- NULL
  for(minCl in minClasses) {
    minExs <- which(data[,tgt] == minCl)
    minExsM <- c(minExsM, minExs)

    if (tgt < ncol(data)) {
      cols <- 1:ncol(data)
      cols[c(tgt,ncol(data))] <- cols[c(ncol(data),tgt)]
      data <-  data[,cols]
    }
```

```
    newExs <- rbind(newExs, smote.exs(data[minExs,], tgt = ncol(data), N = perc.over.r
  }

  selMaj <- sample((1:NROW(data))[-minExsM],
                   as.integer((perc.under/100)*nrow(newExs)),
                   replace=TRUE)

  newdataset <- rbind(data[selMaj,], data[minExsM,], newExs)

  if (is.null(learner)) return(newdataset)
  else do.call(learner, list(form, newdataset, ...))
}



smote.exs <- function(data, tgt, N, k) {
  nomatr <- c()
  T <- matrix(nrow=dim(data)[1], ncol=dim(data)[2]-1)
  for(col in seq.int(dim(T)[2]))
    if (class(data[,col]) %in% c('factor','character')) {
      T[,col] <- as.integer(data[,col])
      nomatr <- c(nomatr,col)
    } else T[,col] <- data[,col]

  if(N < 100) {
    nT <- NROW(T)
    idx <- sample(1:nT, as.integer((N/100)*nT))
    T <- T[idx,]
    N <- 100
  }

  p <- dim(T)[2]
  nT <- dim(T)[1]

  ranges <- apply(T,2,max)-apply(T,2,min)

  nexs <-  as.integer(N/100)
  new <- matrix(nrow=nexs*nT, ncol=p)

  for(i in 1:nT) {

    xd <- scale(T,T[i,],ranges)
    for(a in nomatr) xd[,a] <- xd[,a]==0
    dd <- drop(xd^2 %*% rep(1, ncol(xd)))
    kNNs <- order(dd)[2:(k+1)]

    for(n in 1:nexs) {
      neig <- sample(1:k,1)
```

```
      ex <- vector(length=ncol(T))

      difs <- T[kNNs[neig],]-T[i,]
      new[(i-1)*nexs+n,] <- T[i,]+runif(1)*difs
      for(a in nomatr)
        new[(i-1)*nexs+n,a] <- c(T[kNNs[neig],a],T[i,a])[1+round(runif(1),0)]

    }
  }
  newCases <- data.frame(new)
  for(a in nomatr)
    newCases[,a] <- factor(newCases[,a],levels=1:nlevels(data[,a]),labels=levels(data

  newCases[,tgt] <- factor(rep(data[1,tgt],nrow(newCases)),levels=levels(data[,tgt]))
  colnames(newCases) <- colnames(data)
  newCases
}
```

B Appendix - Additional GLM Output

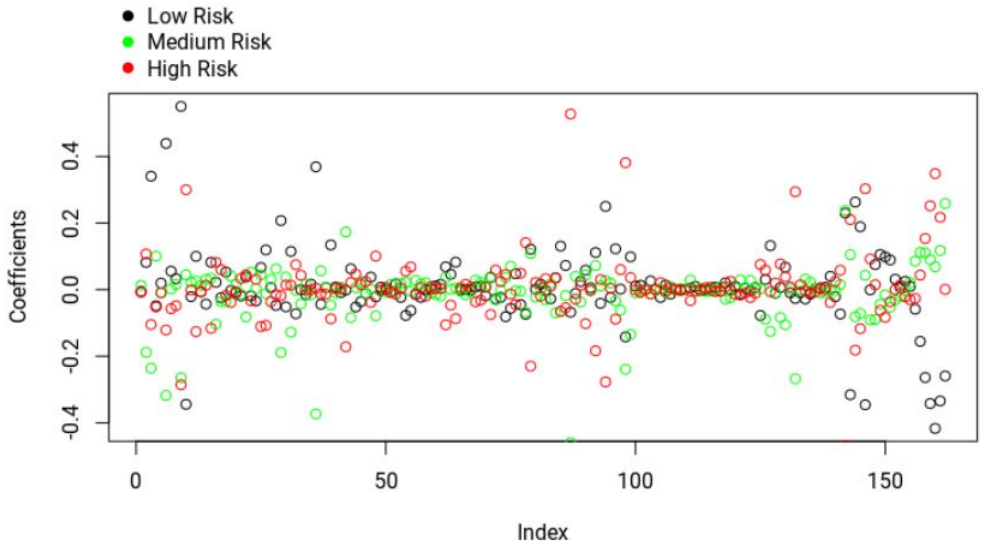


Figure 12: Overview of the GLM coefficients from the second training set.

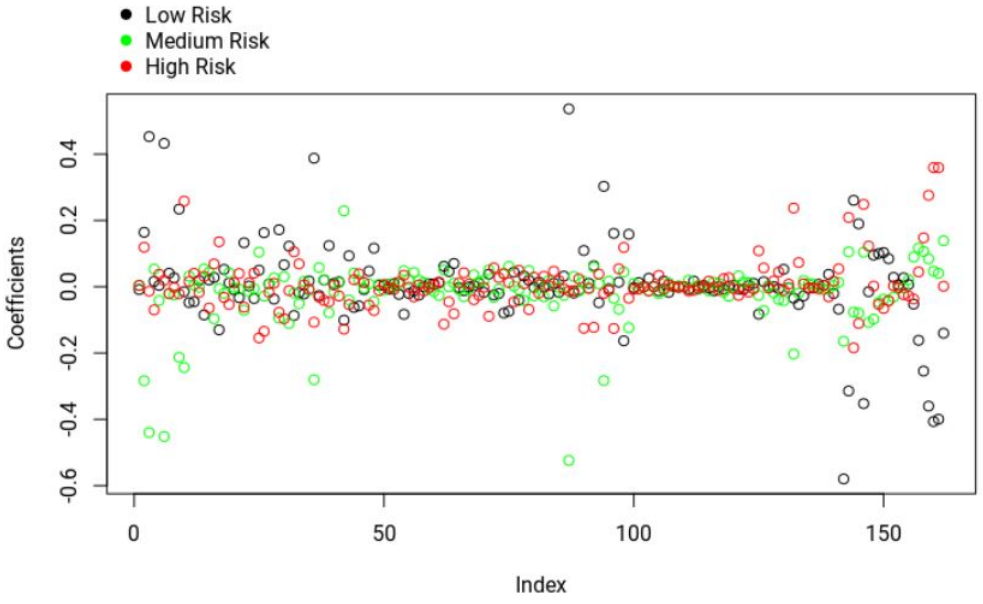


Figure 13: Overview of the GLM coefficients from the third training set.

Table 28: Proportions of risk profiles observed in the top 10 probabilities of belonging to each class in each fold, as computed in GLM.

	Fold 1			**Fold 2**			**Fold 3**		
Top 10	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Top Pr(Low)	10	0	0	10	0	0	10	0	0
Top Pr(Medium)	4	4	2	5	3	2	8	1	1
Top Pr(High)	7	1	2	6	3	1	6	1	3

Table 29: Proportions of risk profiles observed in the top 100 probabilities of belonging to each class in each fold, as computed in GLM.

	Fold 1			**Fold 2**			**Fold 3**		
Top 100	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Top Pr(Low)	98	2	0	99	1	0	99	1	0
Top Pr(Medium)	55	27	18	69	17	14	69	17	14
Top Pr(High)	67	18	15	69	13	18	65	14	21

Table 30: Proportions of risk profiles observed in the top 1000 probabilities of belonging to each class in each fold, as computed in GLM.

	Fold 1			**Fold 2**			**Fold 3**		
Top 1000	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Top Pr(Low)	988	12	0	992	7	1	986	11	3
Top Pr(Medium)	714	181	105	719	192	89	714	177	109
Top Pr(High)	721	163	116	734	156	110	759	133	108

Table 31: AUROC Results under 3-fold Cross-Validation using GLM - Part 1

	$\alpha = 0$					$\alpha = 0.5$					$\alpha = 1$				
Lambda	**Fold 1**	**Fold 2**	**Fold 3**	**SE**	**Average**	**Fold 1**	**Fold 2**	**Fold 3**	**SE**	**Average**	**Fold 1**	**Fold 2**	**Fold 3**	**SE**	**Average**
100	0.62796	0.63080	0.62978	0.00068	0.62951	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
75.4312	0.62805	0.63089	0.62987	0.00068	0.62960	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
56.89866	0.62809	0.63093	0.62991	0.00068	0.62964	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
42.91934	0.62815	0.63099	0.62996	0.00068	0.62970	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
32.37458	0.62822	0.63106	0.63003	0.00068	0.62977	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
24.42053	0.62831	0.63116	0.63013	0.00068	0.62986	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
18.4207	0.62844	0.63128	0.63025	0.00068	0.62999	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
13.89495	0.62860	0.63145	0.63042	0.00068	0.63015	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
10.48113	0.62898	0.63183	0.63079	0.00068	0.63053	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
7.90604	0.62931	0.63216	0.63112	0.00068	0.63086	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
5.96362	0.62973	0.63258	0.63154	0.00068	0.63128	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
4.49843	0.63026	0.63313	0.63208	0.00068	0.63182	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
3.39322	0.63095	0.63381	0.63276	0.00068	0.63251	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
2.55955	0.63179	0.63467	0.63361	0.00069	0.63335	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
1.9307	0.63283	0.63571	0.63464	0.00069	0.63439	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
1.45635	0.63406	0.63695	0.63588	0.00069	0.63563	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
1.09854	0.63549	0.63840	0.63731	0.00069	0.63707	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
0.82864	0.63711	0.64003	0.63893	0.00070	0.63869	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
0.62506	0.63887	0.64181	0.64069	0.00070	0.64045	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
0.47149	0.64072	0.64367	0.64254	0.00070	0.64231	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
0.35565	0.64257	0.64555	0.64440	0.00071	0.64417	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
0.26827	0.64443	0.64743	0.64627	0.00071	0.64604	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
0.20236	0.64620	0.64921	0.64804	0.00071	0.64782	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
0.15264	0.64785	0.65086	0.64966	0.00071	0.64945	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5
0.11514	0.64933	0.65233	0.65111	0.00071	0.65092	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0	0.5

Table 32: AUROC Results under 3-fold Cross-Validation using GLM - Part 2

	$\alpha = 0$					$\alpha = 0.5$					$\alpha = 1$				
Lambda	**Fold 1**	**Fold 2**	**Fold 3**	**SE**	**Average**	**Fold 1**	**Fold 2**	**Fold 3**	**SE**	**Average**	**Fold 1**	**Fold 2**	**Fold 3**	**SE**	**Average**
0.0869	0.65062	0.65361	0.65236	0.00071	0.65220	0.50000	0.50000	0.50000	0.00000	0.50000	0.50000	0.50000	0.50000	0.00000	0.50000
0.0655	0.65173	0.65469	0.65341	0.00070	0.65328	0.50000	0.50000	0.50000	0.00000	0.50000	0.50000	0.50000	0.50000	0.00000	0.50000
0.0494	0.65265	0.65557	0.65427	0.00069	0.65416	0.59143	0.59313	0.59236	0.00040	0.59231	0.50000	0.50000	0.50000	0.00000	0.50000
0.0373	0.65339	0.65627	0.65495	0.00068	0.65487	0.63605	0.63894	0.63684	0.00070	0.63728	0.50000	0.50000	0.50000	0.00000	0.50000
0.0281	0.65397	0.65681	0.65546	0.00067	0.65541	0.63652	0.63941	0.63758	0.00069	0.63783	0.59143	0.59313	0.59236	0.00040	0.59231
0.0212	0.65441	0.65721	0.65584	0.00066	0.65582	0.64111	0.64451	0.64241	0.00081	0.64267	0.63601	0.63890	0.63684	0.00070	0.63725
0.0160	0.65473	0.65749	0.65612	0.00065	0.65611	0.64256	0.64612	0.64392	0.00085	0.64420	0.63638	0.63926	0.63707	0.00071	0.63757
0.0121	0.65496	0.65768	0.65630	0.00064	0.65631	0.64323	0.64683	0.64450	0.00086	0.64485	0.63883	0.64203	0.64004	0.00076	0.64030
0.0091	0.65511	0.65780	0.65642	0.00063	0.65644	0.64505	0.64855	0.64658	0.00083	0.64672	0.64169	0.64499	0.64302	0.00078	0.64323
0.0069	0.65521	0.65787	0.65648	0.00063	0.65652	0.64655	0.65008	0.64806	0.00083	0.64823	0.64263	0.64609	0.64400	0.00082	0.64424
0.0052	0.65527	0.65791	0.65652	0.00062	0.65657	0.64782	0.65137	0.64928	0.00084	0.64949	0.64421	0.64740	0.64543	0.00076	0.64568
0.0039	0.65530	0.65792	0.65653	0.00062	0.65659	0.64878	0.65231	0.65028	0.00084	0.65045	0.64565	0.64906	0.64719	0.00081	0.64730
0.0030	0.65532	0.65792	0.65653	0.00061	0.65659	0.64994	0.65334	0.65135	0.00080	0.65154	0.64707	0.65048	0.64855	0.00081	0.64870
0.0022	0.65532	0.65791	0.65652	0.00061	0.65659	0.65092	0.65416	0.65223	0.00077	0.65244	0.64809	0.65157	0.64954	0.00082	0.64973
0.0017	0.65532	0.65790	0.65651	0.00061	0.65658	0.65174	0.65493	0.65314	0.00076	0.65327	0.64932	0.65272	0.65076	0.00081	0.65093
0.0013	0.65531	0.65789	0.65650	0.00061	0.65657	0.65244	0.65565	0.65392	0.00076	0.65400	0.65042	0.65372	0.65179	0.00078	0.65198
0.0010	0.65531	0.65788	0.65648	0.00061	0.65656	0.65314	0.65626	0.65464	0.00073	0.65468	0.65141	0.65456	0.65276	0.00074	0.65291
0.0007	0.65530	0.65787	0.65647	0.00061	0.65655	0.65380	0.65679	0.65526	0.00071	0.65528	0.65216	0.65532	0.65358	0.00075	0.65369
0.0005	0.65529	0.65786	0.65646	0.00061	0.65654	0.65434	0.65723	0.65570	0.00068	0.65576	0.65286	0.65599	0.65435	0.00074	0.65440
0.0004	0.65528	0.65785	0.65645	0.00061	0.65653	0.65471	0.65752	0.65601	0.00066	0.65608	0.65354	0.65657	0.65500	0.00072	0.65504
0.0003	0.65528	0.65785	0.65644	0.00061	0.65652	0.65496	0.65771	0.65622	0.00065	0.65630	0.65415	0.65706	0.65553	0.00069	0.65558
0.0002	0.65527	0.65784	0.65644	0.00061	0.65652	0.65513	0.65780	0.65636	0.00063	0.65643	0.65458	0.65742	0.65589	0.00067	0.65596
0.0002	0.65527	0.65784	0.65643	0.00061	0.65651	0.65523	0.65786	0.65644	0.00062	0.65651	0.65487	0.65765	0.65614	0.00066	0.65622
0.0001	0.65527	0.65783	0.65643	0.00061	0.65651	0.65527	0.65789	0.65648	0.00062	0.65655	0.65507	0.65778	0.65631	0.00064	0.65639
0.0000	0.65527	0.65783	0.65643	0.00060	0.65651	0.65530	0.65790	0.65649	0.00061	0.65656	0.65519	0.65784	0.65642	0.00063	0.65648

Table 33: An overview of the variables whose effects were consistent across all three folds for the low, medium and high risk profiles respectively. In addition, this table presents which variables were influential in all three, or just two or one of the classes, as an indication of variable importance.

Low	**Medium**	**High**	**All Three**	**Just Two**	**Just One**
Car Variable A	Car Variable A	Car Variable A	Car Variable A	Car Variable B	Car Variable E
Car Variable B	Car Variable D	Car Variable B		Car Variable C	Car Variable H
Car Variable C	Car Variable H	Car Variable C		Car Variable D	Individual Variable C
Car Variable D	Social Variable B	Car Variable F		Car Variable F	Individual Variable D
Car Variable E	Social Variable F	Car Variable J		Car Variable J	Individual Variable F
Car Variable F		Individual Variable B		Social Variable A	
Car Variable J		Individual Variable E		Social Variable D	
Individual Variable B		Social Variable C		Social Variable B	Social Variable E
Individual Variable C				Social Variable C	Social Variable F
Individual Variable D					Social Variable G
Individual Variable E					
Individual Variable F					
Social Variable A					
Social Variable B					
Social Variable C					
Social Variable D					
Social Variable E					
Social Variable G					

C Appendix - Additional Gradient Boosting Output

Table 34: Average AUROC values computed over three fold cross validation over each fold of the training set for various input parameters in gradient boosting.

Interaction Depth	Num. Trees	Shrinkage	Fold 1	Fold 2	Fold 3	Average
5	500	0.01	0.6563	0.6593	0.6579	0.6578
5	1000	0.01	0.6580	0.6600	0.6589	0.6590
5	500	0.001	0.6428	0.6598	0.6584	0.6537
6	500	0.01	0.6564	0.6595	0.6582	0.6581
6	1000	0.01	0.6581	0.6611	0.6593	0.6595
6	500	0.001	0.6444	0.6474	0.6455	0.6458
7	500	0.01	0.6566	0.6597	0.6583	0.6582
7	1000	0.01	0.6581	0.6611	0.6600	0.6597
7	500	0.001	0.6458	0.6486	0.6464	0.6469
8	500	0.01	0.6568	0.6598	0.6584	0.6584
8	1000	0.01	0.6582	0.6612	0.6600	0.6598
8	500	0.001	0.6464	0.6493	0.6474	0.6477

Table 35: Proportions of risk profiles observed in the top 10 probabilities of belonging to each class in each fold, as computed in GBM.

	Fold 1			Fold 2			Fold 3		
Top 10	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Top Pr(Low)	10	0	0	10	0	0	10	0	0
Top Pr(Medium)	4	5	1	2	4	4	4	4	2
Top Pr(High)	7	2	1	8	2	0	6	1	3

Table 36: Proportions of risk profiles observed in the top 100 probabilities of belonging to each class in each fold, as computed in GBM.

	Fold 1			Fold 2			Fold 3		
Top 100	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Top Pr(Low)	99	1	0	99	1	0	100	0	0
Top Pr(Medium)	57	32	11	62	24	14	55	31	14
Top Pr(High)	63	25	12	71	16	13	61	22	17

Table 37: Proportions of risk profiles observed in the top 1000 probabilities of belonging to each class in each fold, as computed in GBM.

	Fold 1			Fold 2			Fold 3		
Top 1000	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Top Pr(Low)	991	7	2	987	9	4	988	11	1
Top Pr(Medium)	702	196	102	681	230	89	677	220	103
Top Pr(High)	702	175	123	712	170	118	707	177	116

D Appendix - Additional Neural Network Output

Table 38: An overview of the AUROC values resulting from 3-fold cross validation over the three folds of the training set.

# Neurons	AUROC Fold 1	AUROC Fold 2	AUROC Fold 3	Average
20	0.584558	0.614599	0.570914	0.5900
30	0.535561	0.500016	0.5486	0.5281
40	0.572239	0.629162	0.535849	0.5791
50	0.607805	0.624621	0.548668	0.5937
60	0.620535	0.637374	0.5401	0.5993
70	0.625037	0.631275	0.535890	0.5974
80	0.539492	0.636487	0.585749	0.5872

E Appendix - Additional Gradient Boosting & SMOTE Output

Table 39: Average AUROC values computed over three fold cross validation over each fold of the training set for various input parameters of gradient boosting trained on MM SMOTE-data.

Depth	Num. Trees	Shrinkage	Fold 1	Fold 2	Fold 3	Average
7	600	0.01	0.7738	0.7757	0.7743	0.7746
7	700	0.01	0.7771	0.7788	0.7775	0.7778
7	800	0.01	0.7797	0.7812	0.7798	0.7802
7	700	0.001	0.7035	0.7067	0.7050	0.7035
8	600	0.01	0.7772	0.7787	0.7774	0.7778
8	700	0.01	0.7799	0.7816	0.7802	0.7806
8	800	0.01	0.7824	0.7837	0.7823	0.7828
8	700	0.001	0.7062	0.7092	0.7019	0.7062
9	600	0.01	0.7794	0.7808	0.7796	0.7799
9	700	0.01	0.7822	0.7834	0.7824	0.7827
9	800	0.01	0.7844	0.7802	0.7802	0.7816
9	700	0.001	0.7086	0.7113	0.7104	0.7101

F Appendix - Neural Networks & MM SMOTE

Table 40: An overview of the AUROC values resulting from the neural networks after 3-fold cross validation over the three folds of the MM SMOTE-data training set.

# Neurons	AUROC Fold 1	AUROC Fold 2	AUROC Fold 3	Average
20	0.500013	0.564560	0.660461	0.5750
30	0.549772	0.564558	0.610920	0.5751
40	0.618216	0.680672	0.551561	0.6168
50	0.663140	0.612050	0.669841	0.6483
60	0.674397	0.687489	0.670363	0.6774
70	0.703266	0.626696	0.670743	0.6669
80	0.665823	0.638779	0.602748	0.6358
90	0.676940	0.624248	0.628994	0.6434

Table 41: The average sensitivities of the neural networks, trained on the MM SMOTE-data, as described in Section 4.3

Variable	Sensitivity
Car Variable D	0.05951
Individual Variable B	0.04170
Car Variable J	0.02770
Car Variable F	0.02676
Car Variable B	0.02514
Car Variable E	0.01491
Individual Variable E	0.01397
Social Variable F	0.00747
Individual Variable F	0.00672
Car Variable G	0.00525
Car Variable C	0.00334
Social Variable G	0.00074
Social Variable C	0.00030
Car Variable A	0.00023
Social Variable B	0.00002
Car Variable I	0
Individual Variable A	0
Car Variable H	0
Social Variable D	0
Individual Variable D	0
Individual Variable C	0
Social Variable E	0
Social Variable A	0

G Appendix - Cost-Sensitive Boosting

Table 42: Average AUC in three-fold cross-validation on one of the three folds for various amounts of trees in the cost-sensitive boosting method.

Number of Trees	**AUC**
50	0.508351
100	0.509406
150	0.508791
200	0.508171
250	0.507210
300	0.507231

H Appendix - Cost-Sensitive Neural Networks

Table 43: The average AUROC resulting from training the cost-sensitive neural network by means of three-fold cross-validation on each of the three folds.

Neurons	**Fold 1**	**Fold 2**	**Fold 3**	**Average**
40	0.5513	0.5571	0.5542	0.5542
50	0.5530	0.5510	0.5505	0.5515
60	0.5734	0.5749	0.5736	0.5740
70	0.5652	0.5689	0.5660	0.5667

Table 44: The average sensitivities, as described in Sections 4.3, of the variables used in the cost-sensitive neural networks.

Variable	**Sensitivity**
Individual Variable B	0.045115
Individual Variable E	0.024902
Car Variable D	0.018431
Car Variable E	0.014315
Individual Variable A	0.003121
Car Variable A	0.002955
Car Variable F	0.00219
Social Variable B	0.002176
Individual Variable F	0.002122
Individual Variable C	0.001781
Social Variable E	0.00083
Car Variable I	0.00054
Individual Variable D	0.00041
Social Variable D	0
Car Variable B	0
Car Variable C	0
Social Variable G	0
Social Variable F	0
Car Variable J	0
Car Variable G	0
Social Variable C	0
Social Variable A	0
Car Variable H	0

I Appendix - Gradient Boosting with Case Weights

In addition to the methods presented in this thesis, gradient boosting was run in combination with case weights. This implies that rather than having a penalty of 1 for each observation, the penalty is dependent on the true class of the observation. The cost of misclassifying a high risk individual is the (anonymised) expected claim size in the high risk group and the same for an individual in the medium risk class. For the low risk group, the cost remains 1 because a cost of 0 (there are no claims in particular year for this group) would distort the outcomes as misclassifying a low risk individual would have no consequences in the objective function.

Given that the cost structure here is unilateral as it depends only on the true class and not on the predicted class, as is the case in the cost-sensitive methods presented above, it is a less sophisticated approach to incorporating costs. However, with an AUROC 0.664634 and expected total costs of 5639918.02, it performs unexpectedly well. Additionally, it has the advantages of providing the same depth of insights of gradient boosting and is much swifter to run than the cost-sensitive variant. An overview of the variable influence as well as the prediction rates are presented below. The parameters used were those found for the stand-alone gradient boosting machine, hence the results may be slightly better if tuned accordingly.

Table 45: Average relative influence of the variables across the three training sets, as found in gradient boosting with case weights.

Variable	**Average Relative Influence**
Individual Variable E	50.133
Individual Variable B	28.624
Individual Variable F	5.289
Car Variable F	4.976
Car Variable A	2.715
Car Variable I	2.109
Car Variable B	1.468
Car Variable E	1.205
Social Variable C	1.025
Social Variable F	0.718
Car Variable H	0.610
Car Variable G	0.413
Car Variable D	0.252
Car Variable J	0.207
Social Variable B	0.176
Car Variable C	0.048
Individual Variable C	0.023
Social Variable G	0.006
Individual Variable A	0.003
Social Variable A	0
Social Variable D	0
Individual Variable D	0
Social Variable E	0

Table 46: Confusion tables for each fold resulting from the trained gradient boosting model with case weights.

	Fold 1			**Fold 2**			**Fold 3**		
	Low	**Medium**	**High**	**Low**	**Medium**	**High**	**Low**	**Medium**	**High**
Low	295378	27519	103644	295011	24934	106718	294010	27326	105164
Medium	11424	3064	11290	11494	2720	11890	11194	3142	11574
High	3131	836	4977	3050	684	4761	3095	842	4915

Table 47: Average proportions of risk profiles observed in the top 10, top 100 and top 1000 probabilities of belonging to each class, as computed in gradient boosting with case weights.

Top 10	**Low**	**Medium**	**High**
Pr(Low)	98.89%	0.93%	0.17%
Pr(Medium)	81.57%	15.03%	3.40%
Pr(High)	76.47%	12.77%	10.77%
Top 100	**Low**	**Medium**	**High**
Pr(Low)	99.33%	0.33%	0.33%
Pr(Medium)	74.67%	22.33%	3%
Pr(High)	74%	12.67%	13.33%
Top 1000	**Low**	**Medium**	**High**
Pr(Low)	96.67%	3.33%	0%
Pr(Medium)	83.33%	16.67%	0%
Pr(High)	70%	10%	20%

J Appendix - Running Times

Table 48: Approximate time required to train the models on one of the three folds. Note that gradient boosting and neural networks in combination with MM SMOTE are simply the same algorithms but trained on a smaller, synthetic data set.

	Approximate Training Time on One Fold
GLM	<1 hour
Gradient Boosting	7 hours
Neural Network	8-9 hours
Cost-Sensitive Gradient Boosting	24 hours
Cost-Sensitive Neural Network	20 hours