University Utrecht

Computing Science

Thesis presented for the degree of Master of Science

Machine Learning in Bankruptcy Prediction

Author:
Frank Wagenmans

Supervisor:
Dr. A. J. Feelders

July 3, 2017

ICA-3870154

Abstract

Bankruptcy prediction is a well researched and important topic that can be important in many decision making processes. In this work we contribute by analysing the predictive strength of payment behaviour data, and applying machine learning techniques on this data in a realistic setting. First, we review the topic of bankruptcy prediction. Second, we introduce the novel type of data available to a pension fund, along with the challenges in structuring the data appropriately. We then develop Logistic Regression, Neural Network, Random Forest, and Decision Tree models. We show that models based on payment behaviour are very well capable of predicting bankruptcy in the next 12 months. We conclude that Random Forests outperform the other techniques, while Logistic Regression models in conjunction with the Elasticnet regularization technique closely follow.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Bankruptcy is the legal status of a corporation that is issued when the entity is unable to repay the debts it owes to creditors. Filing for bankruptcy offers the victim corporation a chance to be forgiven debts that cannot be paid, and it offers creditors a chance to obtain a partial repayment. Bankruptcy filings are often caused by managerial errors, errors in company policy, and the influence of the immediate and general environments of the company [Ooghe and De Prijcker, 2008]. The recent financial crisis has led to a higher number of bankruptcies, which is a prime example of how environment can influence financial health. These financial failures can have a large impact on investors, employees, partners, and even society, and are generally associated with high costs. The need and use of being able to successfully predict a corporation's financial health in some time in the future, is evident. An early warning can help stakeholders minimize the damage, or even completely avert a bankruptcy. This explains the significant attention the topic has gotten in academic literature. The field of bankruptcy prediction has been a well researched topic since several decades. Financial ratios, numbers, and characteristics are analysed to be able to indicate corporate well-being and bankruptcy risk. In the 1930's, research concluded that firms going bankrupt exhibit significantly different financial numbers than successful firms [Smith and Winakor, 1935]. This indicated that further analysis might prove helpful in bankruptcy prediction.

1.1 Related Work

In [Beaver, 1966], one of the first models for predicting bankruptcy is created, through the use of financial ratios. This idea is extended upon in [Altman, 1968], where *Multiple Discriminant Analysis* or MDA is proposed as a well performing technique to aid in bankruptcy prediction. MDA is a method for compressing multivariate data to produce lower-dimensional data suitable for classifica-

tion [Duda et al., 2012], and Altman has created important results for the field of bankruptcy prediction by trying to analyse multiple variables in conjunction instead of sequentially. According to Altman, bankruptcy could be explained quite successfully through the use of five financial ratios: working capital/total assets, retained earnings/total assets, earnings before interest and taxes/total assets, market capitalization/total debt, and sales/total assets. These ratios formed a basic set of firm characteristics informative for corporate distress, that has been widely used in research.

Several shortcomings in Altman's techniques were pointed out and overcome by [Ohlson, 1980], who instead uses a conditional logistic model. The results of both Altman and Ohlson were compared empirically in [Begley et al., 1996], who concludes that the proposed models do not perform well on more recent data, as compared to when the models were originally estimated. However, Begley's results do indicate Ohlson's model to be the most successful in predicting financial distress.

The research up to this point is criticized on one important aspect in [Zmijewski, 1984], in which it is observed that the data used to train the models was never truly realistic. Zmijewski shows that usually data was used where non-bankrupt versus bankrupt companies were evenly distributed, even though (in 1934-1984) only .75% of companies ever went bankrupt. Consistently oversampling the minority class was the chosen solution in this imbalanced class problem, but it was never addressed. This is a significant oversight, since estimating models on such datasets using techniques that assume random sampling might cause estimation bias. This effect will be even stronger when incomplete observations are included in the data-set, and the missing values are interpolated using some heuristic. Zmijewski proposes a weighted regression model, and shows that estimation bias introduced by the oversampling is present. Interestingly, classification rates did not seriously improve.

Research on using neural networks in bankruptcy prediction started around 1990. One of the first attempts is described in [Odom and Sharda, 1990]. Odom and Sharda used Altman's financial ratios to serve as inputs to a neural network with multiple hidden layers, and achieved correct classification rates of around 80%. A comparison is made in [Tam and Kiang, 1992], between logit, k-nearest neighbours, decision trees, and the novel application of neural networks. Tam and Kiang report that the predictive power of the neural network applies to bankruptcy prediction, as it outperforms all other tested techniques. Pompe and Feelders report a case in which this performance difference does not apply, and neural networks do not outperform MDA and decision trees [Pompe and Feelders, 1997]. Neural networks can naturally be used in conjunction or to support other techniques, such as the hybrid approach is described in [Lee et al., 1996]. Lee et al.

proposed a technique in which neural networks were combined with variable selection methods using MDA, ID3, and Self Organizing Maps, and it is shown that the right preprocessing can improve neural network performance. In the following years emerged a wealth of research studies on neural networks, of which [Vellido et al., 1999] provides a clear overview with a section on bankruptcy prediction.

In 2001 a different approach is proposed, by [Shumway, 2001]. The new insights in this article point out the value of taking into account time-dependent variables, instead of using variables at one static point in time. Shumway applies a hazard model, which is a type of survival model in statistics. The model predicts the probability of a firm filing for bankruptcy in some period t, conditional on surviving until the end of the previous period $t-1$ and other variables that are observed at the current period. His results indicate that this class of statistical survival models prove very helpful and promising in assessing corporate risk. This direction of research was extended by [Chava and Jarrow, 2004] and [Campbell et al., 2008], where the time period interval was drastically increased and the forecasting horizon was pushed further.

Some time later, support vector machines, or SVM's were applied to the subject. For example in [Van Gestel et al., 2003], where a Least Squares Support Vector Machine (LS-SVM) was implemented. The LS-SVM performed remarkably well compared to classical techniques, and seemed an effective and simple technique to apply to bankruptcy prediction. A different type of SVM was applied in [Shin et al., 2005], and tested against the generally high performing Neural Networks (NN's). Besides the a gain in transparency and simplicity, the SVM classifier was more accurate than the implemented NN. It needs to be noted that this was mainly the case for relatively smaller data-sets, and the authors point out that training a good NN for these sizes would require more expertise and time.

A different approach from the ones outlined above, is the family of evolutionary computing. This research field, with its roots copied from the biological evolution process, has also received attention in bankruptcy prediction. It might not directly be clear how genetic algorithms can be applied to classification problems, so we have to take a different viewpoint. We need to look at the classification problem as the necessity of one rule, comprised of variables, logical conditions such as greater than, and cut-off values. It then becomes a logical collection of single variable thresholds, that finally outputs a 1 or a 0, or a *bankrupt* or *not bankrupt*. One can view a rule such as this as a string of characters, and this string can be mutated, two of these strings can be combined to produce offspring, one string can be selected over another string due to its performance, and a population of these strings can evolve to one 'best' solution. Note that this is just one interpretation of how we can represent rules for classification as a string. A detailed implemen-

tation of this technique in the field of bankruptcy prediction can be seen in for example [Shin and Lee, 2002], and [Varetto, 1998]. The idea is further explored in [Min et al., 2006], where genetic computing was effectively utilized to aid SVM's in optimizing the feature subset and parameters. In essence, Min created a hybrid form of genetic algorithms and support vector machines, and produced positive results both in SVM tuning and bankruptcy prediction.

The last decade, most research was focused on improving and comparing existing models. A meticulous comparison is made in [Kumar and Ravi, 2007], where a total of 128 scientific papers and more than 500 variables that were used in 1968–2005 were reviewed. Kumar and Ravi note that nearly all statistical techniques have been employed to solve this problem. Their review shows that of all techniques, neural networks were the most popular option. They do however note that a trend was visible in which stand-alone techniques were losing popularity, whereas ensemble or hybrid models were gaining attention and showing improvements in performance. A good example is presented in [Xiao et al., 2012], where the predictive power of logistic regression, neural networks, and support vector machines was combined. The outputs of the three single models were combined into an ensemble model and weighted, and results show that the combined method outperformed each of the three when predicting singularly. This example of the power of hybrid models is more broadly justified in the survey reported in [Verikas et al., 2010]. Verikas et al. compare a collection of hybrid and ensemble methods, where they focus on why and how different techniques were combined. They once again note the power captured in many of the model combinations, at the cost of transparency. They also conclude that the non-linear nature and lack of widely accepted procedures for the design of hybrid or ensemble models form major pitfalls in application of these technologies.

One of the more recent reviews of prediction techniques applied on our area of interest is presented in [Sun et al., 2014]. This article sketches an accurate image of the current state of affairs concerning bankruptcy prediction. The especially interesting part is where the shortcomings and oversights of past researches and results are described. Sun et al. identify several directions of further research, among which is the development of dynamic modelling methods that can cope with gradually emerging samples of data. Another topic is developing imbalanced predictive models, since the real-world situation consists of very imbalanced datasets, although most research focuses on fairly balanced data. The last oversight that was identified, is developing early warning systems. Most research focused on predicting bankruptcy, without considering real-world applicability.

1.2 Research Questions

As we have seen in the previous section, a wealth of research has been conducted in the field of bankruptcy prediction. All well-known techniques have been extensively tested and analysed. As it seems, these experiments were mostly based on taking advantage of some set of financial ratios that describe the status of a company. Altman introduced a basic set of these ratios in [Altman, 1968], and this set has been extended and altered to include many characteristics describing measures such as liquidity and profitability. The performance of using these numbers in bankruptcy prediction is well-documented, but more sorts of data may prove interesting for the issue at hand. In this case, an interesting collection of data becomes available when access is provided to the database of a pension funds' financial data. In this database resides a substantial collection of information that can produce a meaningful insight in payment behaviour, and how this behaviour changes over time. As can be seen in the literature study section, no research was found that actually used this kind of data in bankruptcy prediction. This need not come as a surprise, since payment behaviour is usually stored in data that is well-shielded from outside parties. The novelty of this type of data leads us to the aim of the research, which can be formally stated as:

- *How accurately can we predict bankruptcy using temporal data on payment behaviour?*

Creating a good prediction model is the main aim. As explored in the related work section, a large group of techniques have proven fruitful in bankruptcy prediction. Since there was never a clear winner in terms of best performance, we need to compare different techniques. This leads to the following sub-question:

- *Which technique is the most successful in predicting bankruptcy?*

An important aspect considering real-world applicability is the timespan over which we will be able to predict bankruptcy. If we were to turn the problem at hand into a simple classification problem, it would mean that we try to predict whether a company *has gone* bankrupt. A prediction of this type may be of no practical use to the pension fund whatsoever. We will try to create an early-warning system, which leads us to the following sub-question:

- *How far ahead can we predict bankruptcy?*

Considering the type of data we have, which will be explained in more detail in the next section, we can ask ourselves how much history we want to take into account. It may prove impossible to predict a bankruptcy two or three years ahead, leading one to doubt the predictive characteristics of data from two or three years ago. This leads us to the next sub-question:

- *How relevant is historic data on payment behaviour for bankruptcy prediction?*

Another problem that we will be faced with in this research, is the extreme imbalance of the classification problem. Although methods exist to deal with this complicating aspect of classification tasks, it is a problem that we will need to keep in mind and actively attempt to counter. This leads us to the following sub-question:

- *How can we successfully counter the class-imbalance in the data?*

Lastly, we will look at an important aspect in many experiments in machine learning. This involves the identification, calculation, and combination of characteristics in the data that are predictive of the class label. The last sub-question we will try to answer is therefore:

- *Which properties of payment behaviour over time are predictive for bankruptcy?*

1.2.1 Relevance of the Research

If we look at this research as a whole, the relevance can first be seen when we look at the nature of the field. Predicting bankruptcy in a manageable time-window can help distressed companies to take appropriate measures at an early stage. It can also help creditors take action to prevent (a part of the) loss that can be caused by a bankruptcy. In this specific case, it can help the pension fund to minimize losses. For this party, it can be of utmost importance, since a pension fund is usually low on the priority list of creditors. Enough research has been done in the field of bankruptcy prediction to be able to say that to be of real significance to science, we need to accomplish one or more of the following goals:

- Improve existing techniques
- Invent a novel approach to the field
- Apply existing or new techniques to a new type of data

In this research, a novel approach is tried, whilst using a new type of data. As stated before, no research was found that tried to forecast bankruptcy using 'insider'-data of payment behaviour. Another interesting aspect to take into account is the temporal nature of this dataset. (Which is not completely novel, as the hazard model proposed in [Shumway, 2001] already noted the possibility and promising performance of using multi-period models.) The usage of new types

of data is only one of the directions of further research that were proposed in [Sun et al., 2014]. The other two directions described before were developing models on imbalanced data, and early warning systems. Both of these issues will be addressed in the solution proposed in this paper, as we will see in the following sections.

Chapter 2

The Data

For this work, a large dataset was supplied by the pension fund. The dataset used is comprised of invoice data between the pension fund and 97.671 companies. The set consists of 3.530.000 invoices, which each have identifying columns, the height of the invoice (in Euro), the date it was issued, the date it would expire, and the date it was paid (null if it had not yet been paid). Two columns that may or may not be filled, describe when the most recent overdue reminder was sent (if any), and the 'reminder-level'. This level is a convention kept by the pension fund, which can vary between 0-5, where 0 means that the invoice is not overdue, and 5 is the highest, most critical level. The available invoices are from 2011-2016, where the invoices from before 2014 were issued in an interval of three months, and from 2014 onward they were sent monthly. To accompany the invoice data, we have a set of companies which have gone bankrupt, a total of 675 companies. This set also contains the date on which the company was declared bankrupt.

The last set of data has information on the number of participants a company pays pension contributions for. We have this data at a monthly interval, so it nicely matches the information contained in the invoices.

2.1 Data Cleaning

Some invoices are not bound to companies, which can be seen by looking at the company identification number. If the ID is larger than 100.000.000, the invoice was sent to an individual. These invoices are excluded. Furthermore, invoice identification numbers are recycled every couple of years. This has caused some inconsistent data to show up in the set. We counter this by removing every invoice in which the invoice date exceeds the expiration date, or in which the invoice date exceeds the paid date. Next, we add a bit that indicates whether the invoice has been fulfilled or not, which can be seen by whether the paid date is filled.

Afterwards we take the sum of the number of unpaid invoices per company, so we can use it as a feature. We then remove all unpaid invoices from the dataset, since these cannot be used to calculate other features.

There are cases in which invoices are fulfilled partially. In this case, three new invoices are created: one invoice that has the partial amount that was paid, an invoice with the negative amount of the initial invoice, and then an invoice that shows the leftover amount. When this happens, we have two invoices cancelling each other out, the original one, and the negated version of the original one. These invoices were removed from the set, since they are never actually paid, and thus do not contribute useful information.

We have removed artefacts introduced by partially fulfilled invoices, so all leftover negative invoices are payments from pension fund to company, and they do not imply company payment behaviour. The leftover negatives are thus removed. The resulting collection contains 1.387.320 invoices of 32.568 companies, of which 650 (2%) have gone bankrupt.

2.2 Feature Creation

We start off with a fairly succinct dataset that contains a good amount of information, only accessible when transformed appropriately. An initial idea would be to summarize all invoices per company, leaving us with simple counts and measures that may not be detailed enough. It becomes more interesting when we do not consider the invoices as a heap of information, but as a sequence of information. A sequence in which certain characteristics may change over time. We need to find a full set of features that can: 1) describe overall payment behaviour, and 2) describe the changes in payment behaviour.

With this basic division of feature types in mind, we can continue with the detailed description of the features. We will calculate a set of basic transformations of the source data, which we can than transform into the features of the two described types. The first basic transformation we calculate is called *relatively late*: the number of days an invoice was overdue when it was paid, scaled by the difference between issue date and expiration date. A similar transformation is *late*: simply the number of days an invoice is overdue. Furthermore, we have *number of participants*, *reminder level*, and *amount*.

To correctly calculate the second group of features, we have to take note of the fact that the invoices were sent over two different sizes of intervals. The largest interval is the only one we can work with, which is the 3 month, or quarterly interval. This means we have to somehow aggregate the monthly interval invoices to quarterly intervals. The technique used to do this is called Piecewise Aggregate Approximation (PAA)[Keogh et al., 2001]. PAA decreases dimensionality, by di-

viding the data in equal sized frames. The mean value of the data falling within each frame is calculated, and taken as the new value. It is easy to see that PAA directly applies to the situation of reducing monthly interval invoices to quarterly interval invoices, by creating frames with the length of three months, and taking the average of the variables falling within these frames. Suppose we have three invoices of single company in a quarter year, with the amounts of 100, 170, and 250 euro. The PAA transformation allows us to treat these three invoices as one by taking their average, i.e.: $(100 + 170 + 250)/3 = 140$. This can be done for all numerical variables.

Once this is done, we can continue with the feature creation. We will start off with the first type of features, which describe overall payment behaviour. Suppose again we have a sequence of information for a single company, and we have calculated the basic transformations described above. For the relatively late, reminder level, and number of participants, we add the value of each quarter as a feature. We take the means of the relatively late, late, reminder level, number of participants, and amount variables. We also take sums of the relatively overdue, overdue, and amount variables. We take the range and the variance for the relatively late and number of participants variables. To complete the first set, we add the number of reminders, and the number of invoices. This leaves us with a small set of features that can summarize payment behaviour over any length of time.

The second set is more extensive and complicated, but brings a detailed description of payment behaviour over time. Suppose for example we have 4 quarters of data for a company. For each of these 4 quarters we have the basic transformations, on which we have applied PAA. From the variables *relatively late*, *number of participants*, *reminder level*, and *amount*, we can now calculate differences between every pair of consecutive quarters, in this case amounting to 3 differences for each variable. Since different companies have varying magnitudes of the absolute size of their variables, it seems fair to calculate proportional differences instead of absolute differences. In addition to the 3 differences, we calculate the number of times a difference is positive, the *mean* of the differences, the *variance* of the differences, the *skewness* of the differences, and the *kurtosis* of the differences. For the sake of completeness: *variance* is the average of square differences from the mean, *skewness* is a term that describes the asymmetry of the probability distribution, and *kurtosis* quantifies the 'flatness' of the shape of the distribution.

Lastly, there are two basic features to accompany the ones described above. First is the risk variable, which is the a priori chance that a company of a certain sector goes bankrupt, calculated on historic numbers of bankruptcies in different sectors. Secondly, the days registered with the pension fund, calculated by the difference between the last and first invoice date of a company, of all invoices

present in the selected subset of the data.

To summarize, an overview of all features can be viewed in Table 2.1.

Feature	Description
Relatively Late	Days an invoice was overdue when it was paid, scaled by the difference between issue date and expiration date *(we take the mean, sum, variance, range, and actual value over all quarters)*
Number of Participants	Number of participants a company pays pension contributions for *(we take the mean, sum, variance, range, and actual value over all quarters)*
Reminder Level	Numerical level of severity of payment-reminders that are being sent *(we take the mean, and actual value over all quarters)*
Amount	Amount of the invoice *(we take the mean, variance, skewness, and kurtosis over all quarters)*
Late	Days an invoice was overdue when it was paid *(we take the mean, and sum over all quarters)*
Number of invoices	Number of invoices that were sent to a company
Number of reminders	Number of reminders that were sent to a company
Late invoices	Number of late invoices
Difference relatively late	Proportional difference of 'relatively late' between each pair of subsequent quarters *(we take the mean, variance, skewness, kurtosis, and actual value over all quarters, we also count the number of times a difference is positive/negative)*
Difference number of participants	Proportional difference of 'number of participants' between each pair of subsequent quarters *(we take the mean, variance, skewness, kurtosis, and actual value over all quarters, we also count the number of times a difference is positive/negative)*
Difference reminder level	Proportional difference of 'reminder level' between each pair of subsequent quarters *(we take the mean, variance, skewness, kurtosis, and actual value over all quarters)*
Difference amount	Proportional difference of 'amount' between each pair of subsequent quarters *(we take the mean, variance, skewness, and kurtosis over all quarters)*
Risk	A priori chance that a company of a certain sector goes bankrupt, based on sector numbers
Days Registered	Difference between the issue date of the last and first invoice of a company

Table 2.1: Overview of all features

Chapter 3

Methods

It is common knowledge that different machine learning problems require different approaches, and different machine learning algorithms. We do however not always know in advance which technique is appropriate for the problem at hand. Although the number of algorithms has been growing swiftly, they often share characteristics that allow them to be categorized in several classes. In order to test the vast wealth of algorithms, we can decide to take some of the most common algorithms that cover a large part of the several classes. The techniques that were chosen are presented in this section, along with the performance measure used to quantify the performance of the models that were created.

3.1 Performance Measures

To evaluate the performance of a model, the so called AUC-score is used. AUC, or Area Under the Curve, measures the area under the ROC-curve. The ROC-curve, the Receiver Operating Characteristics curve, is created as follows: we plot the True Positive Rate against the False Positive Rate at some range of thresholds. To provide some insight in how to interpret the AUC: a random classifier should have an AUC of about 0.5, a perfect classifier scores an AUC of 1. Furthermore, the AUC is equal to the probability that the classifier will rank a randomly chosen positive sample above a randomly chosen negative sample, where the ranking is based on the fitted probability of the positive class.[Fawcett, 2006].

The choice for this performance measure can be justified by the fact that the available data has a very uneven class-distribution; as we have seen in the previous section, only 2% of the companies have actually gone bankrupt. This means that if we use simple accuracy as a performance measure, always classifying a sample as not bankrupt gives us the exceptional performing model scoring 98% accuracy. Using AUC instead, we only select models that have True Positive Rates and

False Positive Rates that are significantly better than random chance. Effectively, it discourages choosing representative models that are not discriminative. Another advantage of using AUC is that it is an evaluation of the classifier over a spread of thresholds, instead of evaluating a classifier in conjunction with a chosen threshold.

The AUC scores allows a quick selection of the model that is most able to differentiate between to classes. However, it does not yet select the optimal decision threshold. In order to find the optimal decision threshold, the model can be evaluated by domain experts under different thresholds. The users of the model will have to indicate the relative cost they associate with False Positives and False Negatives, after which the optimal model according to domain expertise can be selected. This research focuses on just the AUC-scores.

3.2 Techniques

This section gives a short overview of the mathematical background needed to understand the algorithms used in this research. We look at how models can be trained, how they can be used to make predictions, and which parameters can be varied to achieve better performance.

3.2.1 Decision Trees

The decision tree is the first and one of the best-known techniques. Decision trees are well known for their exceptional interpretability. The smaller decision trees can be understood by both computer scientists and domain experts, providing a valuable communication means between the two parties. A more sophisticated, less prone to overfitting variant of the decision tree is the random forest. This ensemble approach comes with a loss of interpretability. The ensemble variant will be explored in the next section.

Basic Variant

The basic variant of the decision tree has a tree-like structure, in which every individual node represents some kind of test on one of the attributes of the data. The steps that are taken to predict the class probability using a decision tree are as follows: a test sample falls down tree, taking the left or right branch according to the test criterium, until it ends up in one of the leaves. The leaves of a decision tree all have a prediction probability or class label that is assigned to any test sample that ends up in the leaf in consideration.

Before we can use the tree for prediction, we need to train one. Calculating the test criteria present at the nodes of the tree can be a fairly complex and

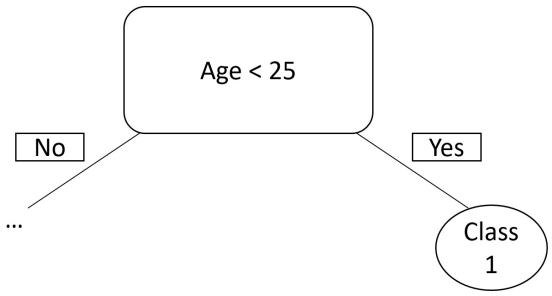

Figure 3.1: Split node in a decision tree

time-consuming process, dependent of the size of your dataset. The process to accomplish this consists of two stages: 1) recursively splitting the training set until some stopping criteria, so we obtain a node and test criteria at every recursion, and 2) assigning the left-over leaves either the class majority or the class probability, calculated over the training samples present in the leaf. The first stage requires some explanation, as it is not directly apparent how the splits are obtained. We will first examine what such a split should actually look like, before we go into the computational aspect.

As can be seen in Figure 3.1, a split at a node sends incoming samples to one of two subsequent branches. It does so based on the criterium present in the node, which is a threshold on one of the attributes of the data. The splitting node divides all training samples that end up in this part of the tree in two regions, that may either be leaves or other splitting nodes. To calculate the best split, we consider all attributes and their possible split-points, and select the setting that produces the child nodes with the lowest *Gini index*:

$$G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{3.1}$$

where $\hat{p}_{mk}$ represents the proportion of training observations in the m-th region of the k-th class. The Gini index takes a value close to 0 when the class proportions are close to 0 or 1. It is therefore a measure of the *purity* of a region.

The algorithm greedily selects best splits, until some criteria is reached. Stopping criteria may describe a minimal threshold of samples present in a node, or a minimal necessary reduction of the *Gini index*. Once all branches in a tree have reached a stopping criteria, we assign either the class probability or class majority to each of the leaves, so we are left with a complete and usable tree.

We need to note that although decision trees exhibit exceptional interpretability, accuracy is often not considered top-shelf. This property can be countered by constructing an aggregation of decision trees, using methods such as *random*

forests, or *boosting*.

3.2.2 Random Forests

A common ensemble variant of the decision tree is the conveniently named *Random Forest*. As the name sketches, it is constructed by semi-randomly creating trees, which together form a forest. In order to produce a prediction, every tree decides on a class. When all trees have chosen, the mode (most common answer), is chosen as the final decision of the forest. The technique is fairly straightforward: calculate some number of trees, each based on a random subset of the features available. The calculated decision trees are not pruned, thus often being more complex than the regular decision tree. As can be imagined, the computational effort of creating such a forest rises linearly with the number of trees it is comprised of. Moreover, the interpretability boasted by the regular decision tree, is mostly lost. The random forest does however minimize overfitting tendencies of its single tree counterpart, and generally a performance increase can be observed.

3.2.3 Logistic Regression

Logistic regression is the method of calculating a best-fitting linear combination of one or more independent variables that can determine an outcome. We will look at how a Logistic Regression model is trained, and how it can be used to predict class labels.

The Basics

Suppose we want to predict a response that falls into one of two classes, 0 or 1. We will not model this response directly, but we will calculate the probability that a sample belongs to a certain class. In scientific terms, we want to predict the chance $P(Y = 1)$. We have some predicting variables, which we will for now call X_1 and X_2. We will thus calculate the probability: $P(Y = 1|X_1, X_2)$. To calculate this value we can use a linear combination of the input variables, where the first one that comes to mind is the *linear regression function*:

$$P(Y = 1|X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \tag{3.2}$$

There is one problem with this approach: we need to output a value between 0 and 1, and the linear regression function can theoretically output any number. We can of course try to alter the various β-values, but there is a better solution. In logistic regression, we will use the *logistic function*:

$$P(X) = \frac{\exp^{\beta_0+\beta_1 X_1+\beta_2 X_2}}{1+\exp^{\beta_0+\beta_1 X_1+\beta_2 X_2}} \tag{3.3}$$

Notice that in equation 3.3 we use $P(X)$ as shorthand for the term $P(Y = 1|X)$ in equation 3.2. To get a prediction for sample x we just need to compute the output of the logistic function. To be able to do this, the values for the collection of β-s need to be estimated. This is where the actual training of the logistic regression model occurs. The regression coefficients are instantiated with an initial solution, and are then slightly revised in an iterative manner, until the improvement of the model becomes too small. What we want to do, is maximize the *likelihood function*:

$$L(\beta_0, \beta_1, \beta_2) = \prod_{i:y_i=1} P(x_i) \prod_{i':y_{i'}=0} (1 - P(x_{i'})) \tag{3.4}$$

Note that we have replaced the logistic function, including the β-values, with P. One can see from this formula that the highest value will be achieved when all samples that have label $y_i = 1$ receive a prediction of (close to) 1, and all samples that have label $y_i = 0$ receive a prediction of (close to) 0. It is generally easier to work with the log of this formula, which is called the *log-likelihood*:

$$\mathcal{L}(\beta_0, \beta_1, \beta_2) = \sum_{i:y_i=1} \log P(x_i) + \sum_{i':y_{i'}=0} \log(1 - P(x_{i'})) \tag{3.5}$$

Maximizing this log-likelihood will also maximize the likelihood, so we can work with this version equivalently. The maximum log-likelihood occurs where its gradient is zero, since the log-likelihood function is globally concave and therefore any local maximum is a global unique maximum. Although we are not going to be able to set the gradient to zero and solve exactly, we can use a numerical optimization method and solve it approximately. The optimization method used in many implementations is the *Fisher-scoring* method, and it is the method that will be used throughout this research. The mathematical details of Fisher-scoring are beyond the scope of this paper, but the interested reader is referred to [Demidenko, 2007].

Since the simple Logistic Regression method is prone to overfitting, many regularization and variable selection methods have been introduced. One of these is called *Elasticnet*, extensively described in [Zou and Hastie, 2005]. This is a combination of the well-known LASSO and Ridge regularization techniques. Due to the widely proven efficacy of the elasticnet technique, it will be applied throughout this research.

3.2.4 Neural Networks

The neural network is another well-known technique that will be applied in this research. The method is named after the network of neurons that is present in a human brain, and it functions in a slightly similar manner. We again consider the case where we want to predict a response that belongs to category 0 or 1.

We have a set of input variables X, consisting of $x_1, \ldots, x_p$. We now construct a network with three layers. The first layer contains nodes for each of the input variables, the second layer contains an arbitrary number of 'hidden' nodes and is therefore called the hidden layer, the third layer contains just one node, called the output node. Note that we can theoretically use any number of hidden layers, but we only consider the single hidden layer variant. Between each of the input nodes and the hidden nodes, we create an arc with an associated weight. We do the same between each of the hidden nodes and the output node. We call the linear combination of weights and values at each of the hidden nodes or the output node the activation of the unit. The activation of hidden unit j, which receives input from units $x_1, \ldots, x_p$, can be more formally written as:

$$a_j = \sum_{i=1}^{p} w_{ij} x_i \tag{3.6}$$

Each a_j is transformed using a non-linear *activation function* to give the actual output value z_j of node j. Since we are performing classification with two classes, the activation function in the output nodes is the logistic sigmoid σ:

$$z_j = \sigma(a_j) = \frac{1}{1 + \exp^{-a_j}} \tag{3.7}$$

The z-values are once again linearly combined into an activation-value, and non-linearly transformed using the logistic sigmoid to produce the output prediction. The process of calculating the values at each layer, and using them to calculate values at the next layer, is called *forward-propagation*. To provide some intuition in the difference between neural networks and logistic regression, it might help to look at a neural network the following way: We essentially 'learn' good features z_j, which are (often unintuitive) combinations of the input values. We then perform logistic regression on the set of z_j's to learn the weights for each of the nodes.

We have seen how we can calculate predictions given a neural network, but how is this network constructed? Consider the above situation, and we want to produce a network with a given number of hidden nodes. The only values we have to optimize, are the weights of the various arcs. To do this, we use the *gradient-descent* optimization algorithm, which essentially takes steps proportional to the gradient

of the error function, to approach a local minimum. We first instantiate the neural network with random weights. We then compute the local error δ of the output neuron, and *back-propagate* this to the hidden neurons, to compute the δ_j's of the hidden neurons. To compute the local error of the output neuron, we use:

$$\delta_y = \hat{y} - y \tag{3.8}$$

Where $\hat{y}$ is the prediction value, or the value that the output neuron produces, and y is the actual class label of a single sample. Note that if we were to compute this over a set of samples, we would have to calculate the sum-of-squares of all of the errors of the individual samples. To compute the local error of the hidden neurons, we use:

$$\delta_j = z_j(1 - z_j)w_{jy}\delta_y \tag{3.9}$$

We then need to compute the gradient of each weight. For the input neurons to the hidden neurons this is:

$$\frac{\partial E}{\partial w_{ij}} = x_i \delta_j \tag{3.10}$$

For the hidden neurons to the output neuron this is computed as:

$$\frac{\partial E}{\partial w_{jy}} = z_j \delta_y \tag{3.11}$$

Once we have the gradient of each weight, we use its value to correct the weights with some learning rate η. We calculate the new weight as:

$$w_{ij}^{k+1} = w_{ij}^{k} - \eta \times \frac{\partial E}{\partial w_{ij}} \tag{3.12}$$

We apply this gradient-descent until the algorithm converges to a minimum, and we have found a solution.

Finding the best model is a somewhat more complex process than in the case of logistic regression. One characteristic of neural networks, is that the error function usually has many local minima, and the basic training algorithms are greedy in nature. As a result, if we initialize our model with a fixed set of weights we may bias our model to go towards a local minimum. To try and overcome this, weights are initialized randomly, which is especially useful when done multiple times in the training process. We can either choose the network created from the set of initial weights that performs best on the training set, or use all networks and average their predictions. Another difficulty of training neural networks is that training the algorithm needs two extra parameters; the decay, and the size of the hidden

layer. The latter one is not too difficult, we can simply try a variety of sizes up to a certain threshold, and select the best performing setting. The decay parameter needs some explanation. It is used in a regularization technique, that causes large weights to be punished to obtain a smooth fit, and avoid over-fitting. The decay parameter, λ, changes the weight correction function:

$$w_{ij}^{k+1} = w_{ij}^{k} - \eta \frac{\partial E}{\partial w_{ij}} - \lambda \eta w_{ij}^{k} \tag{3.13}$$

The decay parameter can also be tried and thus tuned over a range of settings, until the best performing one is found.

Extensions to the basic variant

Neural Networks often require more or different preprocessing than for example tree-based methods. Initial tests proved that simply training a Neural Network on our data was not sufficient. To increase the efficacy of the neural network, several approaches have been applied to our case. The first and most simple step was to pre-process the data differently. A simple and effective preprocessing step is to make sure the data is distributed uniformly [Shi, 2000]. This was done by scaling and centering the data. In other words, we divided each of the variables by their standard deviation, and then subtracted their mean.

Another previously mentioned pitfall in Neural Networks is their tendency to get stuck in local minima. In which local minima a neural network ends up, is possibly dependent on the weights we initialize the nodes and connections with in the training procedure. To reduce this effect, we decided on the strategy of creating an ensemble learner of 10 different neural networks, created by 10 different sets of starting weights. Each of these networks predicts a bankruptcy chance for each record in the testing set. We collect all 10 predictions, and take their average as our definitive chance. In summary, we will apply the basic Neural Network, and additionally: 1) Preprocess the data by scaling and centering, and 2) Decrease variance with an ensemble method.

3.3 Tools

The main tool used for this research is the software environment for statistical computing R[R Core Team, 2016]. The R environment provides a specialized programming language specifically aimed towards statistical computing and graphics. It facilitates fast and efficient data storage and manipulation, and is easily extended with open source-packages. For the machine learning techniques, we used the packages glmnet[Friedman et al., 2010] for Logistic Regres-

sion with Elasticnet, nnet[Venables and Ripley, 2002] for the Neural Networks, randomForest[Liaw and Wiener, 2002] for the Random Forests, and rpart[Therneau et al., 2017] for the Decision Trees. The most important package used in this research is called Caret[from Jed Wing et al., 2016]. The caret package (short for Classification And Regression Training) comes with a set of functions that supports the process of creating predictive models. Many different machine learning algorithms have an interface implemented in this package, to provide a uniform approach for different techniques. Moreover, the process of training and tuning a model is somewhat standardized within this model, relieving the user of common tasks. It also allows the user to calculate the contribution of variables to different kinds of models, providing a simple insight in variable importance. The variable importance is usually a function that is native to the underlying algorithm libraries. For decision trees and random forests, the variable importance is based on the decrease in predictive accuracy when the values of a variable are permuted. In the case of logistic regression, the absolute value of the t-statistic for every variable is used. For neural networks, the importance is calculated by a specialized technique based on [Gevrey et al., 2003], which uses combinations of the values of the weights.

Chapter 4

Experimental Setup

4.1 Setup of the training and testing sets

One aspect is shared throughout the different experiments, which is the way we transform our complete dataset into test and training sets. When we look at the available data, it is not straightforward to imagine a simple division into sets that are suitable for the training and testing of machine learning models. The problem that we are faced with, is that we do not have uniform sets of data for all companies. The data is distributed over several years, where not every company has data available for each time period. We cannot randomly divide all data into two sets, since we then end up with an unstructured and uneven heap of data, on which a model cannot be trained and tested. We are in need of a subset of the data, in which we have a sequence of the same length of data for all the companies present in the subset. Moreover, we want to combine examples of non-bankrupt companies with examples of bankrupt companies.

In the creation of training and testing sets, a set of parameters was used to vary the size and selection criteria of the data. In this section we will cover all parameters that have been invented to vary the form of our training set, allowing different settings we can use to fine-tune the models. Each of the parameters is fully explained when it is encountered in the following paragraphs. An overview of all parameters is presented at the end of this section.

Before we start with how we define the training and testing sets, we first need to fully define our class labels. To achieve this, we have a parameter that tells us how far ahead we will try to predict bankruptcy. This parameter, which we call the *forecastlength*, sets the number of months that we will use in the definition of our class labels. If the forecastlength is set to 3, companies that will file for bankruptcy within 3 months from the end of the test get the class label 1, in any

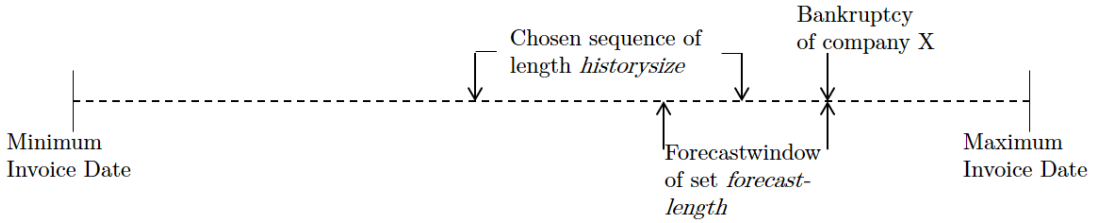

Figure 4.1: Selecting a training sequence for a bankrupt company

other case the class label is set to 0. This means that if we classify a company as bankrupt, with a forecastlength of 3, the classification is only correct if the company actually goes bankrupt in these exact 3 months. Note that this 3 month window is not a set window, as it can slide over our timeline depending on the date we choose to be the date we want to start our predictions from.

4.1.1 Construction of the Training set

To create the set of data for training, we look at all cases of bankruptcy and take some sequence of their data. We need to ensure that all these sequences can be transformed into the same number of features, meaning that they must all have the same number of quarters of data. First, we need to set the parameters that restrain the *timewindow* of the pool of data we can take into account: the *minimum* and *maximum invoice date*. Then, for every bankruptcy that occurred within this timewindow, we look at the bankruptcy date, and go back some months in time to decide on the end of the sequence for the case at hand. The number of months we go back in time is dependent on the the forecastlength-parameter; we choose randomly from $\{1; forecastlength\}$. When we have the ending of the sequence, we go back a set number of quarters to decide on the starting date of the sequence. The number of months we go back is parameterized by the *historysize*, in units of the number of quarters in the sequences that we want to use. The described process to choose a sequence of length *historysize* for a bankruptcy case is visualised in Figure 4.1.

We repeat the process for each bankruptcy that falls in the timewindow between the minimum and maximum invoice date. We then have a sequence with a set starting and ending date, for each bankruptcy. To accompany the bankruptcies, we will add a set number of sequences of non-bankruptcies, with the exact same starting and ending dates. We can vary the proportion of bankruptcy to non-

bankruptcy to fine-tune the models, which we will do by setting the parameter *number of matches* (a setting of 1 will give us 1 non-bankruptcy per bankruptcy). The number of matches can be viewed as a fine-tuning parameter to adjust the class distribution of our training set, effectively countering the class imbalance problem. This approach is a specialized application of undersampling.

4.1.2 Construction of the Test set

The creation of the test set is less complicated. We use the historysize-parameter, and we take one date that we will treat as the time at which we want to use our data to start predicting; the *test date*. Next, we will take a sequence for all companies that have not gone bankrupt before our testing date, which is comprised of the data within the timespan $\{testdate - historysize; testdate\}$. Note that we can only include companies of which we have invoices over the complete timespan, since otherwise we cannot calculate the same set of features for every company. This also means that we cannot compute predictions for companies that started participating too recently. The described process of the selection of data is visualised in Figure 4.2.

Note that it is theoretically possible for a company to be present in both the testing and training set. This is usually not preferred, as it may cause unrealistically positive performance scoring of a model. When we train a model on some data that includes company X, and the exact same sequence of company X is present in the testing set, the better machine learning techniques will often classify company X correct. In our case, there may be a few companies that are present in both sets. However, the sets will never contain a copy of exactly the same sequence. There are a few arguments as to why we have chosen to allow companies to be present in both sets. First, suppose we want to predict a bankruptcy risk for the complete, healthy population. This would be the case when we apply our techniques in a realistic situation. If we would not allow any of the companies in the testing set to be present in the training set, we could only train on samples that have gone bankrupt since all healthy companies are already present in the testing set! This would severely limit the size of the training set, and in most cases training a model would not even be possible. Secondly, the argument that we train a model on company X and use it to predict for company X, may have both positive and negative influence. If we train the model on the basis of company X not going bankrupt, and it does not go bankrupt in the testing set, we may have an advantage. However, it may be the case that this company does go bankrupt in the testing set, creating a disadvantage. Last, the actual occurrence of overlapping sequences of the same company in training and testing set is fairly low. In one experiment there will be several situations in which training and testing set will be completely disjoint, allowing us to analyse the possible effect of any overlap.

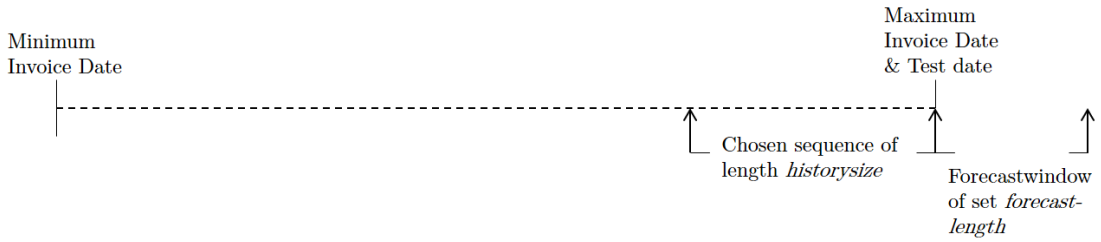

Figure 4.2: Selecting the testing data

The full set of parameters can be seen in Table 4.1.

Parameter	Explanation
Forecastlength	Number of months ahead we will predict bankruptcy
Historysize	Number of quarters of data we include for every sample
Number of matches	Number of non-bankruptcies per bankruptcy in training set
Min. invoice date	Minimum date we allow data to be used from
Max. invoice date	Maximum date we allow data to be used from
Timewindow	Pool of data restricted by min and max invoice date
Test date	End date of the testing set

Table 4.1: All Parameters

4.2 Experiment 1: Relevance of Historic Data

The first experiment revolves around testing the relevance of historic data. We can divide this in two subtly different parts: 1) from how far back should we include data in the training process, and 2) do the characteristics of the data truly change over time. The two aspects were tested in two similar experiments with different set-ups, as can be seen in the next sections.

4.2.1 Experiment 1a: From how far back does data add predictive value?

In short, this experiment will help reason about how far back we should go with adding historic data. By using training sets in which we take into account timewin-

dows that are decreasingly historic, we can simulate a real world application in which we need to decide how much historic data we need to retain. This experiment was initialized by setting the test-date to the first of October, 2015. This was done because we assumed that data from before this date was complete, and invoices that may not have been updated with the most recent information were not included. Next, the forecastlength was set to 6 months. Then the experimental set-up needed a minimum invoice date for the training data, to be able to produce results that can be used to reason about the historical relevance. The minimum date was varied from the end of March 2011, to the end of March 2014, in intervals of three months. By doing this, the timewindow that was used in the training set slowly decreased in size due to leaving out more historical data points. The timewindow defined by the minimum and maximum invoice date was first used to subset the complete dataset. From the subset we continued to select the set of bankruptcies, where bankruptcies were only selected when they occurred within the timewindow. For every setting of the minimum date, the historylength was varied from 3 to 8 quarters, and the number of matches from 1 to 10. The last setting means we vary the proportion of bankruptcies to non-bankruptcies from $\{1:1; 1:10\}$. Per setting of the minimum date, 6 settings of historylength, and 10 settings of number of matches were tested, amounting to a total of 60 training sets per minimum invoice date setting. Note that for the more recent minimum dates not all settings of historylength were viable, since the timewindow became too small to be able to select longer sequences.

The created training sets were used to train a model using 5-fold cross-validation to determine the optimal complexity parameters. The optimal models were then tested on a set with the said test-date of the end of September 2015, and a historysize with the same setting as that of the training set. This experiment was repeated for each of the 4 machine learning algorithms described in the 'Methods' section. The experiment produced results of models trained on sequences selected from data within the timewindow between a minimum and maximum invoice date, where the minimum date increased, the sequences had varying historylength, and varying class distribution (number of matches).

4.2.2 Experiment 1b: To what extent do the predictive characteristics change over time?

The previous experiment used a timewindow of varying size that we could select sequences from. In other words, we shifted the minimum date of the timewindow onward, but we did not alter the maximum date. To further isolate the relevance of historic data, this experiment included a shifting maximum date as well. Structur-

ing our different training sets in this manner, allows us to analyse the predictive performance of data from different time periods, thus providing insight in how much the data changes over time. This was done by using a timewindow with a length of 9 quarters, which was shifted forward over the timeline. The maximum date was varied from the end of September 2013 to the end of September 2015, in intervals of three months. The minimum date was computed as 9 quarters before the maximum date, which allowed for a bit of movement space when we are selecting sequences within the timewindow. Initial tests proved that taking shorter length timewindows simply omitted too many instances to build a training set extensive enough to calculate a model from. This time the historysize was not varied; it was set to 4 quarters. It was decided not to vary this parameter, since it would introduce difficulty in creating enough settings for the timespan, especially when the historysize was large. The number of matches was again chosen from $\{1; 10\}$.

The created training sets were used to train a model using 5-fold cross-validation to determine the optimal complexity parameters. The optimal models were then tested on a set that had the test-date of the end of September 2015, and a historysize of 4 quarters. This experiment was repeated for each of the 4 machine learning algorithms described in the 'Methods' section. This experiment produces results of models trained on datasets of a shifting timewindow, using sequences of historylength 4 and varying class-distribution (number of matches).

4.3 Experiment 2: How far ahead can we predict bankruptcy?

This experiment was mainly performed to find out how far ahead bankruptcy can be predicted based on our data. This was done by varying the forecastlength parameter between 3 months, 6 months, 9 months, and 12 months. In this experiment the testing date was set to the end of September 2015. The maximum training date was also set at the end of September 2015. A minimum training date was not included, meaning the full dataset was always accessible for this experiment. The number of matches was varied from 1 to 10, and the historysize from 3 to 8 quarters.

The created training sets were used to train a model using 5-fold cross-validation to determine the optimal complexity parameters. The optimal models were then tested on a set with the said test-date of the end of September 2015, and a historysize set the same as that of the training set. This experiment was repeated for each of the 4 machine learning algorithms described in the 'Methods' section. Results were produced of models trained on data taken from the complete dataset (no restricting timewindow), using sequences of increasing historylength and varying

class-distribution (number of matches).

4.4 Further analysis in the experiments

First, a closer look was taken at the two parameters that defined the representation of the training and testing datasets; the *historysize* and *number of matches*. We took a closer look at how these parameters influenced performance, producing insights in how well different settings performed and how they should be fine-tuned. Results for the analysis of these observations can be found under Section 5.4.

Furthermore, an aim of this work is to find out the properties of payment behaviour that provide predictive value in bankruptcy prediction. This was analysed by taking the best performing models, and using the caret built-in function `varImp()`. This function returns a score per variable, indicating the relative contribution it adds to the model. It produces a nice overview of the most important variables, under different settings. Results for the analysis of these observations can be found under Section 5.5.

Lastly, recall that one of the aims of this work is to find out what techniques are successful in predicting bankruptcy. In order to reason about their performance, the above described experiments were performed using all 4 machine learning algorithms that were presented in the 'Methods' section. This allowed us to observe the performance of the different algorithms under different circumstances. A comparison was made to decide which is the best fit in the current scope. The analysis of these observations can be found in Section 5.6.

Chapter 5

Experimental Results

In this chapter the results of the experiments will be discussed. Each of the sections covers one of the five experiments described in Chapter 4. It would be unnecessary to report all results, since simply too many parameter settings have been tested. Therefore a succinct overview will be given for each experiment, highlighting the most important findings.

5.1 Experiment 1a: Relevance of historical data - incremental timewindows

In this experiment, several different timewindows of training data with the same maximum invoice dates were tested over a variation of the historysize and number of matches parameters. This produced 60 models per technique and timewindow, each associated with an AUC score. The mean and maximum AUC scores over the 60 models were computed for each of the 4 techniques and 7 timewindows. Note that for the three most recent settings of the minimum invoice date, not all historysizes were tested, so less than 60 models were trained. The maximum scores can be seen in Table 5.1.

The maximum scores show a trend in which the start of the timewindow shifts to the right and becomes smaller, and the performance declines. Only the Random Forests and Logistic Regression seem stable, apart from the small drop when the minimum invoice date was set at 31-3-2014. This small drop may be attributed to the fact that the only historysize tested here was 3 quarters. The same may be true for the Decision Tree in the last case, it was not able to distinguish the cases better than assigning random class labels. All other results seem to show that the more data we have the better the model becomes, which is only natural in machine learning. It becomes more interesting when we combine this observation with the fact that all 'added' data is more and more historical, seemingly showing a fairly

Min.invoice date	*Logistic Regression*	*Neural Network*	*Random Forest*	*Decision Tree*
31-3-2011	0,9115	0,8731	0,9266	0,8745
30-9-2011	0,8978	0,8856	0,9293	0,8449
31-3-2012	0,8929	0,8915	0,9259	0,8435
30-9-2012	0,9079	0,8965	0,9202	0,8230
31-3-2013	0,9099	0,8762	0,9236	0,7733
30-9-2013	0,8919	0,8845	0,9218	0,7327
31-3-2014	0,8690	0,8440	0,8872	0,5

Table 5.1: Experiment 1a - Maximum AUC scores

stable data behaviour over the period of several years.

Min.invoice date	*Logistic Regression*	*Neural Network*	*Random Forest*	*Decision Tree*
31-3-2011	0,8633	0,7838	0,9033	0,5862
30-9-2011	0,8545	0,8070	0,8991	0,5933
31-3-2012	0,8532	0,8166	0,8888	0,6268
30-9-2012	0,8351	0,8052	0,8801	0,6141
31-3-2013	0,8342	0,8099	0,8860	0,5872
30-9-2013	0,8503	0,8401	0,8837	0,5999
31-3-2014	0,8338	0,7887	0,8780	0,5

Table 5.2: Experiment 1a - Mean AUC scores

The mean AUC scores can be seen in Table 5.2. Interestingly enough, the mean scores do not always seem to follow the same downward trend with decreasing timewindow sizes. Apart from the random forests, the mean AUC scores are often far worse than the best AUC scores, indicative of high variance between the results obtained with different parameter settings. Random forests are clearly the most accurate models of experiment 1a.

5.2 Experiment 1b: Relevance of historical data - timewindows of the same length

This experiment tested the 4 techniques on training sets based on timewindows of the same length, varying only in recency. The maximum and mean AUC scores

were computed over the various settings for the parameter number of matches. The best AUC scores can be seen in Table 5.3.

Max.invoice date	*Logistic Regression*	*Neural Network*	*Random Forest*	*Decision Tree*
30-9-2013	0,8996	0,8607	0,9157	0,7345
31-12-2013	0,8960	0,8728	0,9052	0,7754
31-3-2014	0,8990	0,8340	0,9172	0,5077
30-6-2014	0,8772	0,8274	0,9169	0,8417
30-9-2014	0,8555	0,8568	0,9116	0,8060
31-12-2014	0,8854	0,8571	0,9134	0,7901
31-3-2015	0,8863	0,8440	0,9075	0,8108
30-6-2015	0,9082	0,8284	0,9131	0,7575
30-9-2015	0,8447	0,8079	0,9023	0,7226

Table 5.3: Experiment 1b - Maximum AUC scores

The best AUC scores show that Random Forest models again perform best, closely followed by Logistic Regression. Decision Trees seem to be most affected by the differing timewindows, shown by a high variance in scores. Neural networks perform reasonably well, and Decision Trees have consistently low scores. There is no clear pattern to observe in the performance of models trained on increasingly recent data, contrary to the expectation of a slightly increasing performance trend.

Max.invoice date	*Logistic Regression*	*Neural Network*	*Random Forest*	*Decision Tree*
30-9-2013	0,8743	0,8051	0,9003	0,5411
31-12-2013	0,8731	0,8282	0,8753	0,6046
30-3-2014	0,7300	0,7678	0,8831	0,5007
30-6-2014	0,7503	0,7726	0,8865	0,6002
30-9-2014	0,7278	0,8099	0,9066	0,5652
31-12-2014	0,8606	0,8181	0,8923	0,6404
30-3-2015	0,8327	0,8042	0,8752	0,5759
30-6-2015	0,8685	0,7969	0,8745	0,5616
30-9-2015	0,7553	0,7524	0,8873	0,5820

Table 5.4: Experiment 1b - Mean AUC scores

The mean AUC scores for this experiment can be seen in Table 5.4. These results tell the same story as the best AUC scores; the relative recency of the training

data does not matter in our scope. The best and mean scores for random forests are again closest, indicating this technique to be affected least by the changing number of matches. In general, results of experiment 1a and 1b seem to show that historical data is just as relevant as recent data, at least in our case where we only have access to 5 years of data.

5.3 Experiment 2: How far ahead can we accurately predict bankruptcy

This experiment tested 4 different settings of forecastlength, and a variation of other parameter settings. For each of the 4 machine learning techniques, per forecastlength-setting, we computed the mean and maximum AUC score over all other parameter settings. The maximum scores can be seen in Table 5.5.

Forecastlength	*Logistic Regression*	*Neural Network*	*Random Forest*	*Decision Tree*
1 quarter	0,9783	0,9419	0,9839	0,8754
2 quarters	0,9106	0,8844	0,9255	0,8737
3 quarters	0,9111	0,8833	0,9249	0,8689
4 quarters	0,9078	0,8942	0,9252	0,8620

Table 5.5: Experiment 2 - Maximum AUC scores

Looking at the best AUC scores, it is interesting to see that performance does not significantly diminish when we look further into the future. Clearly, predicting only one quarter ahead is most precise, but stay similar with the three larger forecastlengths. Again, random forests are consistently better than the other techniques, closely followed by Logistic Regression with Elasticnet regularization.

Forecastlength	*Logistic Regression*	*Neural Network*	*Random Forest*	*Decision Tree*
1 quarter	0,9311	0,8576	0,9611	0,5518
2 quarters	0,8656	0,7951	0,8995	0,5933
3 quarters	0,8823	0,8155	0,9115	0,6402
4 quarters	0,8819	0,8517	0,9063	0,6900

Table 5.6: Experiment 2 - Mean AUC scores

The mean AUC scores can be seen in Table 5.6. These mean scores over all other parameter settings again show no large differences between the different forecastlengths. In the case of decision trees, performance even seems to increase when the forecastlength is increased. In the case of Logistic Regression, Random Forests, and Neural Networks, the predictive performance does show a drop after 1 quarter. Random forests seem to be the models that have a mean score closest to the maximum score, indicating that this technique is least susceptible to the variations in historysize and number of matches.

5.4 Influence of *historysize* and *number of matches*

It became clear from the previous experiments that the parameters historysize and number of matches for the data preparation and selection had a hand in the performance of the models. The best performing setting for the historysize may tell us something about the amount of history per company we need to incorporate in the model, and thus to what extent data from for instance 1.5 years back actually contributes to prediction accuracy. The other parameter, number of matches, regulates the class distribution in the training set. Observing the influence of varying this parameter allows us to reason about the optimal class distribution for training the best model. We will explore both of these parameters and how they have influenced results in the previous experiments.

5.4.1 Historysize

We first analysed the results of experiment 1a. We have previously shown only the best and the mean AUC scores over all possible parameter combinations. In this section we will only look at the difference in performance when the historysize was varied. A table was created in which the mean AUC score was presented per setting of historysize, per algorithm. The overview can be seen in Table 5.7.

In the cases of Logistic Regression, a slight downward trend can be observed when the historysize is increased. This effect might be attributed to the addition of more historical information to a single case, which might only muddle the predictive capacity of Logistic Regression. However, notice that with increasing historysize, the number of samples decrease slightly, since not all samples have many quarters of data present. We will not explore this effect further, since the parameter is mostly viewed as a condition which is applied to the training set creation, where the consequences to the size of the training set are also of interest. The slight decrease in performance is not truly the case for any of the other algorithms, although variations in performance can be observed. In spite of the fact that there are some shifts in performance, it cannot be said that increasing the

Historysize (quarters)	*Logistic Regression*	*Neural Network*	*Random Forest*	*Decision Tree*	*Mean*
3	0,8736	0,7890	0,8980	0,5679	0,7654
4	0,8443	0,8138	0,8883	0,5970	0,7591
5	0,8571	0,8334	0,8913	0,5962	0,7636
6	0,8437	0,8225	0,8911	0,6296	0,7636
7	0,8310	0,7915	0,8934	0,5786	0,7339
8	0,8246	0,7832	0,8789	0,6424	0,7395

Table 5.7: Mean AUC scores per historysize, Experiment 1a

historysize decreases the performance of tree-based algorithms and Neural Networks. If we would have to select the optimal setting for this parameter, it would be 3 quarters. Naturally, this is the best choice because it allows for the best AUC scores. But increasing the historysize has two other, more subtle effects. First, an increased historysize translates to increased dimensionality. Every quarter added to the history of a company adds 6 variables to the data. Remember that for the variables *relatively overdue*, *number of participants*, and *reminder level*, we add the proportional difference between two subsequent quarters, and the actual value at each quarter as features. Another important point, is that with a lower historysize we lower the requirements for samples to be able to participate in the training set. If a sample only has 4 quarters of data, it will not be added in a training set that requires 8 quarter of data to be available. So in addition to increased performance, a lower historysize introduces lower computational effort, and less data loss. The same analysis was performed for experiment 2. The results are similar, and will thus not be explored. For these particular results, the interested reader is referred to Appendix A.1. Experiment 1b did not incorporate varying historysizes.

5.4.2 Number of matches

We will now look at the influence of the parameter that regulates the class distribution in the training set. To this end, an overview was created in which we present the mean AUC score per setting for number of matches, per algorithm. This analysis was first performed for experiment 1a. The results can be seen in Table 5.8.

These results show that the number of matches setting does not necessarily influence the performance of the models greatly. One may say that there is no clearly optimal setting for Neural Networks, Logistic Regression, and Random Forests, all distributions of bankruptcy to non-bankruptcy perform equally well. The Decision Trees do seem to slightly suffer from a larger class imbalance. Since

Number of Matches	*Logistic Regression*	*Neural Network*	*Random Forest*	*Decision Tree*	*Mean*
1	0,8493	0,8005	0,8891	0,6502	0,7632
2	0,8523	0,8116	0,8915	0,6353	0,7702
3	0,8493	0,8041	0,8875	0,6195	0,7702
4	0,8505	0,8172	0,8916	0,5941	0,7576
5	0,8519	0,8079	0,8891	0,5780	0,7411
6	0,8656	0,8188	0,8878	0,5683	0,7571
7	0,8422	0,8108	0,8839	0,5542	0,7503
8	0,8324	0,8077	0,8852	0,5465	0,7397
9	0,8440	0,7925	0,8881	0,5868	0,7477
10	0,8468	0,8021	0,8904	0,5350	0,7369

Table 5.8: Mean AUC scores per number of matches, Experiment 1a

a higher setting of number of matches means a larger training set, one might be inclined to keep it low. The same analysis was performed for experiments 1b and 2. The results are similar, and will therefore not be explored. The interested reader is referred to Appendix A.2 and A.3.

5.5 Which properties prove predictive for bankruptcy

In this experiment we will explore the influence of various features that were calculated from the data source. In order to do this, we selected the best performing models for experiments 1a, 1b, and 2, per machine learning technique. Afterwards, we applied the built-in Caret library `varImp()`-function, to get a ranking of the most contributing variables of each model. We selected the top 5 contributing variables per model. Since the Neural Networks models consist of an ensemble of 10 networks, we took the top 5 contributing variables of all top 10 networks, and then selected the 5 most frequent variables as the top 5 contributing variables of the ensemble model An overview of the rankings can be found in Appendix Section A.2.

A quick glance at the rankings show mostly *late*-related variables. Moreover, we can often observe *reminder level*-, *number of participants*-, and *risk*-variables in the top 5 rankings. To have a rough estimation of the most important variables over all the best performing models, we scored each of the top 5 variables with the AUC-score that the model in consideration scored. We added the scores together, resulting in a score for each of the variables present in the top models. Since we considered 12 models in this analysis, we considered the variables that were present

in the top 5 rankings of at least 4 of the best models as the most important ones. A ranked table of these 7 variables can be seen in Table 5.9.

Rank	Variable
1	last_relatively_late
2	variance_relatively_late
3	risk
4	mean_late
5	sum_late
6	range_relatively_late
7	range_numberofparticipants
8	mean_relatively_late
9	sum_relatively_late

Table 5.9: Top variables present in at least 4 of the 12 best models.

Interestingly enough, most of these are variables that are transformations from either *late* or *relatively late*. It seems that with a combination of explanatory variables that characterise the punctuality of payment behaviour, we have a solid base for a bankruptcy prediction model. The top variable, 'last_relatively_late', quantifies the relative lateness of payment of the most recent data, and it should not come as a surprise that this property proves very important in bankruptcy prediction.

5.6 Performance of the different techniques

We can see from the results of the previous experiments the random forests consistently outperform the other 3 techniques. In the case of decision tree models, this is not an unexpected result. The decision trees seem unable to capture the complex relations between the features in the dataset.

Logistic Regression, paired with the Elasticnet regularization and variable selection technique, follows the performance of Random Forests closely. Neural Networks, combined in an ensemble, have shown reasonable performance. Compared to Random Forests and Logistic Regression, they are however less favourable. In addition to decreased performance, training 10 Neural Networks is naturally inferior to the other techniques in terms of computational effort.

Summarizing, Random Forests are the best choice in this research, when we only look at performance. Given the fact that Logistic Regression models are more easily interpreted by domain experts, one may be inclined to choose that technique.

Chapter 6

Conclusions

This research focused on predicting company failures, and to what extent this could be done by examining and analysing payment behavioural data. Results have shown that this type of data, after many different transformations, prove to be predictive for bankruptcy. We have used many different preparations, subset selections, and different techniques to tackle the classification problem, producing promising results in the field of bankruptcy prediction. In addition to overall performance results, we have shown results that indicate the most effective machine learning techniques, the most predictive characteristics of payment behaviour, and how far ahead we can accurately predict bankruptcy.

Our results have shown that bankruptcy can be predicted quite accurately, given the correct technique and parameter settings. AUC scores in the range of 0.85-0.95 can be expected under optimal circumstances. Random Forests have proven most effective in this problem, closely followed by Logistic Regression models in conjunction with Elasticnet regularization. Random Forests were least influenced by different parameter settings and required less fine-tuning to achieve near optimal performance, and are therefore the most appropriate machine learning technique in the prediction problem at hand. Decision Trees seemingly lacked the capability of capturing the more complex relations present in the payment behavioural data.

We have shown that we can accurately predict bankruptcies that occur within a year. Experiments pointed out that when we try and identify bankruptcies within a quarter of a year, performance is best. Performance was only slightly decreased when we looked ahead half a year, which was the basis for most experiments. The accuracy of our models did not seem to diminish when we increased the prediction window from half a year to a full year.

We have shown that in order to predict bankruptcy, the relevance of data did not change significantly over the course of 5 years. Better results were achieved when we included less recent data in the dataset used for training the model. More-

over, there were no large performance differences between models trained on only older data and models trained on only more recent data. The payment behaviour indicative of healthy and unhealthy companies seems to have been constant over the time that has passed in the dataset. Furthermore, we have shown that it is not necessary to include a long sequence of data for companies to be able to reason about their financial health. In most cases, including a year of data for each company is enough. Data from more than a year before a bankruptcy did not seem to add predictive information. In some cases a larger historysize even decreased performance, although this may be attributed to the slightly decreased training sets.

We have successfully countered the class-imbalance problem in the dataset for bankruptcy prediction. Training a model on the data as is was not feasible. By controlling the ratio of bankruptcy to non-bankruptcy cases in the training set with the parameter named *number of matches*, it was shown that with more evenly distributed data better results were achieved. In order to have this approach perform optimally, we matched sequences of bankrupt cases with a number of sequences of non bankrupt cases in the exact same time.

The characteristics of payment behaviour that proved most predictive were identified by ranking the variables of the top scoring models based on their contribution. The analyses showed various transformations of the lateness in payment behaviour were often very important in all machine learning techniques. In addition, metrics that characterise lateness and its change over time contributed greatly to the best performing models.

6.1 Future Work

We have successfully created a prototype that is ready to use in production environment. In order to incorporate this in the processes present in the pension fund, we will need to automate data collection, cleaning, transforming, and model learning. The parameters we have introduced, and the consequences of altering their values, will have to be analysed together with domain experts to determine their optimal setting. Furthermore, automated decision making based on bankruptcy risk predictions provided by the model needs to be incorporated in the stream. This is planned to be implemented in the nearby future.

In the light of improving performance, the model can be paired with accounting numbers describing the financial position of companies. These numbers have been the focus of many other researches in bankruptcy prediction, and it will be a good exercise to try and add their predictive capabilities to our descriptions of payment behaviour. It may also be interesting to see how well our models hold against models trained on just financial ratios.

Bibliography

[Altman, 1968] Altman, E. I. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The journal of finance*, 23(4):589–609.

[Beaver, 1966] Beaver, W. H. (1966). Financial ratios as predictors of failure. *Journal of accounting research*, pages 71–111.

[Begley et al., 1996] Begley, J., Ming, J., and Watts, S. (1996). Bankruptcy classification errors in the 1980s: An empirical analysis of altman's and ohlson's models. *Review of accounting Studies*, 1(4):267–284.

[Campbell et al., 2008] Campbell, J. Y., Hilscher, J., and Szilagyi, J. (2008). In search of distress risk. *The Journal of Finance*, 63(6):2899–2939.

[Chava and Jarrow, 2004] Chava, S. and Jarrow, R. A. (2004). Bankruptcy prediction with industry effects. *Review of Finance*, 8(4):537–569.

[Demidenko, 2007] Demidenko, E. (2007). *Mixed Models-Theory and Applications*. JSTOR.

[Duda et al., 2012] Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

[Fawcett, 2006] Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

[Friedman et al., 2010] Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22.

[from Jed Wing et al., 2016] from Jed Wing, M. K. C., 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. (2016). *caret: Classification and Regression Training*. R package version 6.0-73.

[Gevrey et al., 2003] Gevrey, M., Dimopoulos, I., and Lek, S. (2003). Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological modelling*, 160(3):249–264.

[Keogh et al., 2001] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001). Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286.

[Kumar and Ravi, 2007] Kumar, P. R. and Ravi, V. (2007). Bankruptcy prediction in banks and firms via statistical and intelligent techniques–a review. *European journal of operational research*, 180(1):1–28.

[Lee et al., 1996] Lee, K. C., Han, I., and Kwon, Y. (1996). Hybrid neural network models for bankruptcy predictions. *Decision Support Systems*, 18(1):63–72.

[Liaw and Wiener, 2002] Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

[Min et al., 2006] Min, S.-H., Lee, J., and Han, I. (2006). Hybrid genetic algorithms and support vector machines for bankruptcy prediction. *Expert systems with applications*, 31(3):652–660.

[Odom and Sharda, 1990] Odom, M. D. and Sharda, R. (1990). A neural network model for bankruptcy prediction. In *1990 IJCNN International Joint Conference on neural networks*, pages 163–168.

[Ohlson, 1980] Ohlson, J. A. (1980). Financial ratios and the probabilistic prediction of bankruptcy. *Journal of accounting research*, pages 109–131.

[Ooghe and De Prijcker, 2008] Ooghe, H. and De Prijcker, S. (2008). Failure processes and causes of company bankruptcy: a typology. *Management Decision*, 46(2):223–242.

[Pompe and Feelders, 1997] Pompe, P. and Feelders, A. (1997). Using machine learning, neural networks, and statistics to predict corporate bankruptcy. *Microcomputers in Civil Engineering*, 12(4):267–276.

[R Core Team, 2016] R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[Shi, 2000] Shi, J. J. (2000). Reducing prediction error by transforming input data for neural networks. *Journal of Computing in Civil Engineering*, 14(2):109–116.

[Shin et al., 2005] Shin, K.-S., Lee, T. S., and Kim, H.-j. (2005). An application of support vector machines in bankruptcy prediction model. *Expert Systems with Applications*, 28(1):127–135.

[Shin and Lee, 2002] Shin, K.-S. and Lee, Y.-J. (2002). A genetic algorithm application in bankruptcy prediction modeling. *Expert Systems with Applications*, 23(3):321–328.

[Shumway, 2001] Shumway, T. (2001). Forecasting bankruptcy more accurately: A simple hazard model*. *The Journal of Business*, 74(1):101–124.

[Smith and Winakor, 1935] Smith, R. F. and Winakor, A. H. (1935). *Changes in the financial structure of unsuccessful industrial corporations*. University of Illinois.

[Sun et al., 2014] Sun, J., Li, H., Huang, Q.-H., and He, K.-Y. (2014). Predicting financial distress and corporate failure: A review from the state-of-the-art definitions, modeling, sampling, and featuring approaches. *Knowledge-Based Systems*, 57:41–56.

[Tam and Kiang, 1992] Tam, K. Y. and Kiang, M. Y. (1992). Managerial applications of neural networks: the case of bank failure predictions. *Management science*, 38(7):926–947.

[Therneau et al., 2017] Therneau, T., Atkinson, B., and Ripley, B. (2017). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-11.

[Van Gestel et al., 2003] Van Gestel, T., Baesens, B., Suykens, J., Espinoza, M., Baestaens, D.-E., Vanthienen, J., and De Moor, B. (2003). Bankruptcy prediction with least squares support vector machine classifiers. In *Computational Intelligence for Financial Engineering, 2003. Proceedings. 2003 IEEE International Conference on*, pages 1–8. IEEE.

[Varetto, 1998] Varetto, F. (1998). Genetic algorithms applications in the analysis of insolvency risk. *Journal of Banking & Finance*, 22(10):1421–1439.

[Vellido et al., 1999] Vellido, A., Lisboa, P. J., and Vaughan, J. (1999). Neural networks in business: a survey of applications (1992–1998). *Expert systems with applications*, 17(1):51–70.

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

[Verikas et al., 2010] Verikas, A., Kalsyte, Z., Bacauskiene, M., and Gelzinis, A. (2010). Hybrid and ensemble-based soft computing techniques in bankruptcy prediction: a survey. *Soft Computing*, 14(9):995–1010.

[Xiao et al., 2012] Xiao, Z., Yang, X., Pang, Y., and Dang, X. (2012). The prediction for listed companies' financial distress by using multiple prediction methods with rough set and dempster–shafer evidence theory. *Knowledge-Based Systems*, 26:196–206.

[Zmijewski, 1984] Zmijewski, M. E. (1984). Methodological issues related to the estimation of financial distress prediction models. *Journal of Accounting research*, pages 59–82.

[Zou and Hastie, 2005] 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.

Appendix A

Results

A.1 Influence of *historysize* and *number of matches*

Historysize (quarters)	*Logistic Regression*	*Neural Network*	*Random Forest*	*Decision Tree*	*Mean*
3	0,8920	0,7633	0,9231	0,6258	0,7952
4	0,8757	0,8201	0,9261	0,6107	0,7889
5	0,8899	0,8512	0,9129	0,5873	0,7784
6	0,8762	0,8519	0,9184	0,6074	0,7818
7	0,8770	0,8481	0,9147	0,6446	0,7854
8	0,8853	0,8450	0,9222	0,6372	0,7785

Table A.1: Mean AUC scores per historysize, Experiment 2

Number of Matches	*Logistic Regression*	*Neural Network*	*Random Forest*	*Decision Tree*	*Mean*
1	0,8158	0,7914	0,8889	0,6858	0,7493
2	0,8286	0,8026	0,8864	0,6181	0,7438
3	0,7770	0,7972	0,8843	0,6566	0,7519
4	0,8379	0,7910	0,8960	0,5618	0,7219
5	0,7977	0,8031	0,8902	0,5541	0,7247
6	0,8390	0,8004	0,8536	0,5470	0,7217
7	0,7562	0,7932	0,9017	0,5502	0,7054
8	0,8070	0,7905	0,8887	0,5469	0,7163
9	0,8153	0,8090	0,8899	0,5022	0,7105
10	0,8061	0,7717	0,8882	0,5236	0,7066

Table A.2: Mean AUC scores per number of matches, Experiment 1b

Number of Matches	*Logistic Regression*	*Neural Network*	*Random Forest*	*Decision Tree*	*Mean*
1	0,8767	0,8143	0,9098	0,7867	0,8051
2	0,8832	0,8370	0,9112	0,6990	0,7965
3	0,8776	0,8234	0,9199	0,6856	0,7947
4	0,8852	0,8375	0,9195	0,6307	0,7909
5	0,8839	0,8457	0,9233	0,5903	0,7821
6	0,8761	0,8121	0,9188	0,5703	0,7697
7	0,8886	0,8428	0,9235	0,5504	0,7771
8	0,8854	0,8224	0,9202	0,5696	0,7778
9	0,8859	0,8292	0,9221	0,5493	0,7761
10	0,8843	0,8349	0,9273	0,5565	0,7770

Table A.3: Mean AUC scores per number of matches, Experiment 2

A.2 Which properties prove predictive for bankruptcy

	Logistic Regression	*Neural Network*
AUC	0,9115	0,8965
Rank		
1	risk	range_relatively_late
2	skewness_differences_mean_reminderlevel	last_relatively_late
3	range_numberofparticipants	kurtosis_differences_reminder_level
4	last_difference_numberofparticipants	risk
5	late_invoices	second_last_relatively_late

Table A.4: Top 5 contributing variables in best models of Experiment 1a - Logistic Regression & Neural Network

	Random Forest	*Decision Tree*
AUC	0,9293	0,8745
Rank		
1	last_relatively_late	mean_late
2	variance_relatively_late	sum_late
3	range_relatively_late	mean_relatively_late
4	mean_late	sum_relatively_late
5	variance_differences_relatively_late	variance_relatively_late

Table A.5: Top 5 contributing variables in best models of Experiment 1a - Random Forest & Decision Tree

	Logistic Regression	*Neural Network*
AUC	0,9082	0,8728
Rank		
1	risk	positive_differences_numberofparticipants
2	range_numberofparticipants	range_numberofparticipants
3	mean_differences_numberofparticipants	kurtosis_differences_numberofparticipants
4	mean_differences_reminderlevel	positive_differences_relatively_late
5	last_difference_numberofparticipants	risk

Table A.6: Top 5 contributing variables in best models of Experiment 1b - Logistic Regression & Neural Network

	Random Forest	*Decision Tree*
AUC	0,9172	0,8417
Rank		
1	last_relatively_late	sum_relatively_late
2	sum_late	sum_late
3	sum_relatively_late	mean_late
4	mean_relatively_late	last_relatively_late
5	variance_relatively_late	mean_relatively_late

Table A.7: Top 5 contributing variables in best models of Experiment 1b - Random Forest & Decision Tree

	Logistic Regression	*Neural Network*
AUC	0,9783	0,9419
Rank		
1	risk	last_relatively_late
2	last_differences_numberofparticipants	mean_late
3	range_numberofparticipants	sum_late
4	mean_differences_relatively_late	sum_relatively_late
5	skewness_differences_relatively_late	kurtosis_differences_relatively_late

Table A.8: Top 5 contributing variables in best models of Experiment 2 - Logistic Regression & Neural Network

	Random Forest	*Decision Tree*
AUC	0,9839	0,8754
Rank		
1	variance_relatively_late	last_relatively_late
2	last_relatively_late	variance_relatively_late
3	range_relatively_late	range_relatively_late
4	mean_differences_relatively_late	mean_late
5	variance_differences_relatively_late	sum_late

Table A.9: Top 5 contributing variables in best models of Experiment 2 - Random Forest & Decision Tree