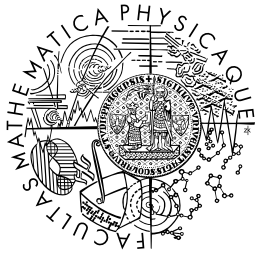

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Bc. Adéla Mrázková

Comparison of statistical methods for the scoring models development

Department of Probability and Mathematical Statistics

Supervisor of the master thesis: Sebastiano Vitali, Ph.D.
Study programme: Mathematics
Study branch: Financial and Insurance Mathematics

Prague 2018

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

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

In Prague date May 8, 2018 Adéla Mrázková

Title: Comparison of statistical methods for the scoring models development

Author: Bc. Adéla Mrázková

Department: Department of Probability and Mathematical Statistics

Supervisor: Sebastiano Vitali, Ph.D., Department of Probability and Mathematical Statistics

Abstract: The aim of this thesis is to introduce and summarize the process of scoring model development in general and then basic statistical approaches used to resolve this problem, which are in particular logistic regression, neural networks and decision trees (random forests). Application of described methods on a real dataset provided by PROFI CREDIT Czech, a.s. follows, including discussion of some implementation issues and their resolution. Obtained results are discussed and compared.

Keywords: credit risk scoring models logistic regression neural networks decision trees random forests

I would like to sincerely thank to my thesis adviser Sebastiano Vitali for his guidance and support. Also I would like to thank to PROFI CREDIT Czech, a.s. company for providing the dataset and especially to Monika Papoušková, who helped me with data understanding.

Contents

Introduction

Credit risk management and credit scoring in particular is very important in the financial field. Various institutions, especially those which offer loans, need to handle credit risk. Credit scoring methods are a good way how to proceed, as we can find it in [Siddigi, 2006]. There are many ways how to build a credit scoring models. The classical approaches are using logistic regression [Hosmer and Lemeshow, 2000] or neural networks [Fyfe, 2000], one could also say, that use of random forests [Breiman, 2001] is a classical approach, but this is usually not done in practice due to limited interpretability.

Just before we start talking about scoring models, let us briefly summarize what is credit risk. It is a risk, that a borrower will fail to repay a debt, i.e. he/she will default. Various institutions need to handle credit risk since they offer products like loans, mortgages, etc. It is clear, that this is very important, because lending money to people without checking their ability to repay the loan can lead to bankruptcy very easily. The crucial issue is how to check this ability.

There are of course many ways how to handle credit risk. One of them is to use a credit scoring model.

A credit scoring model is a tool used to evaluate risk connected with clients, in particular it provides us with information about chance that client will not default. In the other words, it gives us statistical odds that a specific client with particular score will be "good" or "bad". What exactly means "good" and "bad" will be discussed in Section 1.1.1.

This information is typically based on characteristics of a client, such as age, salary, number of kids and many other, selected from any of the sources of data available. To process these characteristics in order to get the information we use various statistical tools. Some of them will be described in Section 1.2.

The aim of this thesis is to introduce and describe the three mentioned approaches (logistic regression, neural networks and random firests) as well as the scoring process itself, since the use of statistical methods is just a part of the whole process. The main steps of this process are definition of default, data exploration, selection of characteristics we use, construction of the model and model implementation and finally reporting. Then these methods are applied and compared on a real dataset provided by PROFI CREDIT Czech, a.s., a company based in the Czech Republic that specializes on providing non-banking loans.

In the first chapter the reader can find overall summary of the whole scoring model development process; first a general description of the whole development process is included, then logistic regression methods, neural network methods and decision trees (random forests) methods are introduced and described.

In the second chapter we describe and discuss some practical implementation issues, including their resolution. These are definition of good and bad client, selection of characteristics, binning approaches, some rules of thumb for model parameters selection, models comparison and early stopping.

Finally, in the third chapter we empirically apply the introduced methodologies to the real dataset provided by PROFI CREDIT Czech, a.s. We show evidences to validate the quality of the proposed algorithm and we suggest possible insights to improve the application of the three methods in the specific

contest.

The main contributions of this thesis are firstly to give overall insight into the scoring models development problematic including the mathematical tools and secondly to show an original comparative study of basic methods on a real consumer loan dataset.

1. Theory

1.1 General model development

Let us introduce the main parts of general model development process. These are:

- Preliminary analysis - Section 1.1.1
- Data Analysis - Section 1.1.2
- Initial characteristic analysis - Section 1.1.3
- Construction of scoring model - Section 1.1.4
- Implementation - Section 1.1.5
- Reporting - Section 1.1.6

1.1.1 Preliminary analysis

Let us now look at the development process. Before we start with the scoring model development itself, several other issues need to be solved. These are:

1. Identify and prioritize objectives.
2. Create project plan, identify project risks.
3. Look at the data carefully.
4. Define the meaning of bad and good client.

Only after we solve these, we can continue with the scoring model development. We will now look at them more precisely.

To solve Issue 1 means to understand what we want from the model. We can require increment of profitability, minimization of losses, better predictive power and many other things, so we need to know what is the most important. We need to have clear decision rules to choose the model which suits our requirements best. Especially competing issues like increasing revenue vs. decreasing losses must be discussed. Typically, the final requirement will be a mixture of more objectives. It is also important to be aware of what is the expected role of our model, i.e. should it be a solely used tool, or a supplement of other methods? Sometimes it is also necessary to decide, whether it is better to develop the model in-house or by an external vendor. This decision depends on factors such as resource availability, expertise in scorecard development, time frame for internal and external development, etc.

Let us now look at Issue 2. Creating a project plan includes clearly defined timelines, implementation strategy, deliverables and should address ideally all important issues. Since the development process is fully reliant on the data we have, there are many risks connected with the quality of data. These are for example insufficient data, dirty or unreliable data, difficulties in accesing data or

nonpredictive data. Ideally all project risks and factors that can potentionally affect the quality of model should be identified at this stage.

Issue 3 is partly connected with Issue 2, since part of the project risks is connected with data quality. We also need to look at data quantity, but this requirement varies. Data quantity needed depends for example on the definition of good and bad client. But in general, it should fulfill requirements of statistical significance and randomness, i.e. it contains no recognizable patterns or regularities. It is important to look at the source of the data. Application data items which are not verified are susceptible to being misrepresented. On the other hand, items such as credit bureau data are robust and can be used. It is necessary to go through the data and exclude all accounts that have abnormal performance, such as staff, VIPs, lost/stolen cards etc., and also frauds. These should not be a part of any development sample. It is also necessary to look whether there might be some effects of seasonality in the data, so that in some period we do not have the "normal" population of clients. This is very important, since scorecards are developed under assumption that the future performance will reflect the past performance. If there are some, we need to diagnose the source of this abnormality and then somehow handle it, for example filter out this source via excluding the problematic part of the data from the development sample.

Issue 4 is crucial in the development. It is clear, that the definition of "bad" affects the final decision rule a lot. We would like to categorize account performance into three groups: "good", "bad" and "indeterminate". The "bad" definition must of course be in line with organizational objectives. A "tighter" definition provides a more extreme differentiation, but can yield to too low sample sizes. On the other hand, a "looser" definition will yield to higher sample sizes, but a weak scorecard [Siddigi, 2006]. The definition must be easily interpretable. There may be some external (for example regulatory) requirements on how "bad" is defined. After the definition is identified, further analysis can be done in order to confirm it. To make sure, that those identified are indeed truly bad. This can be done by expert assessment or analytically. Once we define what is "bad", we need to define also what is "good" and "indeterminate". How to define "good" is usually quite obvious. It is good to notice, that while the "good" status needs to be retained during the whole time period, the "bad" status is usually defined by reaching the specified delinquency stage at any time. "Indeterminate" are those accounts which can not be classified as either "good" or "bad".

There is one more thing connected with Issue 4. It is selection of performance and sample window. Since scoring models are developed under assumption that "the future will be the same as the past", previously opened accounts are analyzed in order to predict performance of future accounts. In order to do this, we need a sample of accounts opened during a particular time period and then to observe their performance during another specific time period. The time period during which we observe the performance of these accounts is called performance window. The time period from which we take sample of already known good and bad accounts is called sample window.

1.1.2 Data Analysis

We have already touched problem of data in Subsection 1.1.1. Now we will look at it more precisely, since this is very important phase. We will discuss the following:

1. Segmentation.
2. Selection of characteristics that will be included in the development sample.
3. Splitting the sample into development and validation datasets.
4. Data collection.
5. Offset method, sampling weights.
6. Exploring data.

Let us start with Point 1. It can happen, that it is more efficient to use slightly different models for distinct parts of the population in dataset than to use one overall scoring model. It is in case that the population consists of subpopulations. This can be checked up statistically or based on experience knowledge. The process of identifying subpopulations is called segmentation and must be always done carefully, it must always be reasonable and the difference between the groups must be always translated into measurable effect on business. Experience-based segmentation can be done for example based on demographics, product type or applicant type. Statistically-based segmentation methods are for example clustering or decision trees. Clustering is a method for identifying groups that are similar to each other with respect to the input variables. It can be performed on the basis of Euklidean distances from one or more quantitative variables. It is good to notice, that the groups are similar to each other based on their characteristics, not performance. On the other hand, decision trees isolate segments based on performance criteria.

Concerning Point 2, selection of characteristics is a critical point of the process. It is done with respect to factors such as expected predictive power, interpretability, reliability and robustness, future availability etc. Some business thought in every stage of the development is needed at this point.

The dataset needs to be divided into development and validation sample, as mentioned in Point 3. Development sample is the sample on which the model is developed and is usually 70% or 80% of the whole dataset. Validation sample is a sample on which the model is validated, i.e. checked if it really works. If the dataset is not big enough to use just 80% for development, the whole 100% of the dataset can be used and the model is then validated using randomly selected samples of 50% to 80%. Typically, about 2 000 of each goods, bads and rejects are sufficient for the development, but there are various statistical techniques to determine the optimal sample size.

Let us now briefly make a few notes refering to data collection (Point 4). While collecting data, we have to be careful about data segmentation. We also need the data to be random and representative, not skewed for example to one region. A very painful problem is data quirks. That is a change of data format during observed period.

Point 5 refers to the situation when the sample population does not reflect the situation in the real population, that is the bad rate in our sample is different from the real one. Then adjusting for prior probabilities is needed and it can be done for example using offset method or sampling weights. The offset method is used in the logistic regression model and is based on shifting the logit. Method of sampling weights is based on multiplying each case by a weight to make the sample reflect true population.

Exploring data (Point 6) is a crucial point of the analysis. It is very important to fully understand all aspects of the dataset. We should make some basic descriptive plots to see what the dataset looks like. And we need to deal with missing values and outliers. We have basically three options how to deal with missing values: we can remove them from the dataset, we can use them as a variable and we can fill them using statistical methods. What to do depends on particular situation and we should be always careful about this.

1.1.3 Initial characteristic analysis

The first step is to assess the strength of each characteristic individually. The strongest characteristics are then binned (grouped). It is possible to produce a scoring model using continuous characteristics, but binning them has many advantages, such as easier way to deal with outliers, nonlinear dependencies can be modeled using linear models and so forth.

Once we are done with the first step, variable selection needs to be done.

The first step in this analysis should be ordering characteristics by some statistical strength measure. These are for example weight of evidence (WOE), which is based on the log of odds calculation, or information value (IV). After we do this, we need to check whether the order is logical or not. The characteristics should be always ordered in a logical way. If there is something illogical, usually helps experimenting with groupings. We should also consider business and operational relevance.

1.1.4 Construction of scoring model

The next step is construction of scoring model using methods described in further sections. This includes:

1. Preliminary model creation.
2. Reject inference.
3. Final model production.
4. Choosing a model and validation

Preliminary model creation (Point 1) means, that we produce one or more models using just a set of approved loans. We follow technique described in Sections 1.1.1, 1.1.2 and 1.1.3. Then we construct the scoring model using selected method, for example we can use one of the methods described in Section 1.2.

Reject inference (Point 2) is the way how we treat rejected loans. The usual way how to proceed in practice is that we just simply ignore them. But there

are several more options, such as assuming that all the rejected loans are bad, which is obviously very strong assumption especially in the case we are not very confident that the previous rule according to which the loans were rejected was reliable. Another option is to produce a model on approved loans, use this model to assign defaults to the rejected loans and then include them into the dataset and build the model again. This is probably better, but still not perfect, since there is risk of adding some "artificial dependencies" which are originally not there into the dataset.

After we decided how to treat rejected loans, we need to produce the final model. Point 3 refers to the situation when we decide not to ignore rejected loans and include them into the development process. In such situation we update the preliminary model or models trained in Point 1 by including rejected data in the way we decided in Point 2.

In the situation when we develop more than one model, we need to choose which one suits us the best. Then the model needs to be validated, that means we check if it behaves reasonably. Some metrics of model behavior quality are introduced in Section 1.3.

1.1.5 Implementation

There are two ways of understanding Implementation in the context of credit scoring model development.

The first one is the implementation of selected mathematical method. That means we take a preprocessed dataset and we apply the selected algorithm in order to obtain a model which suits our requirements. This will be discussed further in Section 1.2 (description of mathematical methods and algorithms which can be possibly used) and Chapter 2 (computational issues).

The other one is implementation in the sense that once we have the model, we need to decide how exactly to use it and then set it working. This means for instance setting the cutoff, i.e. minimum score level at which we are willing to accept applicants and defining the exact way of use of the model, sometimes multiple models are used sequentially or in some kind of combination. For more details regarding this topic please see [Siddigi, 2006].

1.1.6 Reporting

After selecting the final model, management reports are produced. They are good for making operational decisions and monitoring further model performance. For example, it is:

1. Gains table.
2. Characteristic reports.

A gains table includes a distribution of total, good, and bad cases by individual scores or score ranges. The key information in gains table is the expected bad rates for each score or score range, the expected bad rates for all applicants above a certain score and expected approval rates at each score.

Characteristic reports provide distributions for each characteristic included in the model.

Reports done after implementation are good for monitoring model and portfolio performance. Most of these reports are connected with model and portfolio performance statistics. Examples are stability report, model characteristic analysis report, final score report or override report. Since it is not in scope of this thesis to discuss this in detail, for more information see [Siddigi, 2006].

1.2 Model mathematical formulation

As introduced in Section 1.1 we need to select a proper model to evaluate risk of being bad for each client. Therefore, we introduce and describe 3 approaches which are logistic regression, neural networks and decision trees (random forests). We also introduce and describe independence model and weight of evidence model, since they can be understood as a base and motivation of the logistic regression model. The purpose of this section is to highlight the most important features and fundamental ideas of these methods and also some specific extensions of the classical methods which are used in this thesis, such as adaptive learning rate method for neural networks. For a complete description of the models and for other alternative please see for instance [Safavian and Landgrebe, 1990], [Fyfe, 2000], [Hosmer and Lemeshow, 2000], [Breiman, 2001], [Siddigi, 2006].

1.2.1 Independence model, WOE model

Independence model and WOE model are prabably the simplest models used in credit scoring. Let us now define some basic terms.

Definition 1. *[Kozmík, 2006] Let us define odds as*

$$odds = \frac{|B|}{|G|}$$

where B is a set of all defaulted clients and G is a set of all non defaulted clients. Suppose that characteristics of a client are represented by I variables of J_i categories. Then for category $j \in J_i$ of variable $i \in I$ define

$$odds_j^i = \frac{|B_j^i|}{|G_j^i|}$$

where B_j^i and G_j^i are corresponding sets of defaulted and nondefaulted clients, respectively. The odds ratio is then defined as

$$OR_j^i = \frac{odds_j^i}{odds}$$

Definition 2. *[Kozmík, 2006] Let us define the odds function as*

$$odds(\mathbf{x}) = \frac{\mathbb{P}[Y_x = 1]}{\mathbb{P}[Y_x = 0]}$$

where Y_x is a random variable with alternative distribution with parameter $\pi(\mathbf{x})$ and $\mathbf{x}$ is a vector of characteristics of a client.

Note that the parameter $\pi(\mathbf{x})$ has the meaning of probability of default of client with characteristics $\mathbf{x}$ and $Y_x = 1$ if the client with characteristics $\mathbf{x}$ defaulted.

Independence model Having definitions 1 and 2, we now define a simple scoring function:

$$S^{IM}(\mathbf{x}) = odds \prod_{(i,j)\in\mathbf{Z}} (OR_j^i)^{x_j^i} \tag{1.1}$$

where $\mathbf{x} = \{x_j^i, (i,j) \in \mathbf{Z}\}$ is a vector of dummy variables representing the characteristics of a client and $\mathbf{Z}$ is a set of all ordered pairs of variables i and their categories j.

When we look at the function carefully, we can see, that it can be interpreted as an estimate of $odds(\mathbf{x})$ of a client with characteristics $\mathbf{x}$ under the assumption of independence of the characteristics. This assumption is the biggest disadvantage of this approach. Another disadvantage is, that all the variables have the same weight.

WOE model The Weight of Evidence (WOE) model is a generalization of the Independence model described above. To all variables $i \in I$ we assign a weight λ_i according to their statistical importance. We obtain the following scoring function:

$$S^{WOE}(\mathbf{x}, \lambda) = odds \prod_{(i,j)\in\mathbf{Z}} (OR_j^i)^{\lambda_i x_j^i} \tag{1.2}$$

Let us now introduce the definition of Weight of Evidence.

Definition 3. *[Siddigi, 2006] The Weight of Evidence is defined as*

$$WOE_i = \log\left(\frac{\pi_i}{1-\pi_i}\right)$$

where π_i is a probability that client having the property i will be bad.

Note that the quantity in the logarithm is *odds* of characteristic i.

1.2.2 Logistic regression

The logistic regression model can be understood as a generalization of the WOE model. In the WOE model, weights are assigned to each variable. In logistic regression model, weights are assigned to each category of each variable. Then we obtain scoring function of the following form:

$$S^{LR}(\mathbf{x}, \lambda) = odds \prod_{(i,j)\in\mathbf{Z}} (OR_j^i)^{\lambda_j^i x_j^i} \tag{1.3}$$

This can be easily transformed into

$$\log(S^{LR}(\mathbf{x}, \lambda)) = \log(odds) + \sum_{(i,j)\in\mathbf{Z}} \lambda_j^i x_j^i \log(OR_j^i) \tag{1.4}$$

The quantity on the left side of this equation is called logit. Parameters in this model are estimated using the maximum likelihood theory.

Definition 4. *[Hosmer and Lemeshow, 2000] Let $\pi(\mathbf{x})$ be probability of default of a client with characteristics $\mathbf{x}$. Then by logit we mean the quantity*

$$\log\left(\frac{\pi(\mathbf{x})}{1-\pi(\mathbf{x})}\right)$$

Till now, we were looking at the logistic regression as at a generalisation of WOE model. Let us now see it also from the perspective of generalized linear models. The model described above can be rewritten in the form

$$\log\left(\frac{\pi(\mathbf{x})}{1-\pi(\mathbf{x})}\right) = \beta_0 + \beta_1 \cdot x_1 + \cdots + \beta_n \cdot x_n$$

where $\pi(\mathbf{x}) = \mathbb{E}[Y|\mathbf{x}]$, Y is the response, $\beta = (\beta_0, \ldots, \beta_n)$ is vector of unknown parameters and $\mathbf{x} = (x_1, \ldots, x_n)$ is a vector of clients characteristics. That means, that it is a generalized linear model with logit link. That means, that the probability of default itself is modeled by the logistic function.

Definition 5. *[Hosmer and Lemeshow, 2000] The logistic function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as*

$$f(\mathbf{x}) = \frac{\exp(\beta^\top \cdot \mathbf{x})}{1 + \exp(\beta^\top \cdot \mathbf{x})}$$

where $\beta \in \mathbb{R}^n$ is vector of parameters.

Let us explain the interpretation on an example of one independent variable X with possible outcomes 0 and 1. Then, according to our model,

$$\pi(1) = \frac{\exp(\beta_0 + \beta_1)}{1 + \exp(\beta_0 + \beta_1)}$$

$$\pi(0) = \frac{\exp(\beta_0)}{1 + \exp(\beta_0)}$$

$$1 - \pi(1) = \frac{1}{1 + \exp(\beta_0 + \beta_1)}$$

$$1 - \pi(0) = \frac{1}{1 + \exp(\beta_0)}$$

Then the odds ratio is

$$OR = \frac{\frac{\pi(1)}{1-\pi(1)}}{\frac{\pi(0)}{1-\pi(0)}} = \frac{\frac{\frac{\exp(\beta_0+\beta_1)}{1+\exp(\beta_0+\beta_1)}}{\frac{1}{1+\exp(\beta_0+\beta_1)}}}{\frac{\frac{\exp(\beta_0)}{1+\exp(\beta_0)}}{\frac{1}{1+\exp(\beta_0)}}} = \exp(\beta_1)$$

which means, that the outcome is approximately $\exp(\beta_1)$ times more likely to be present (equal to 1) for individuals with $x = 1$ than with $x = 0$.

Since it is not in scope of this thesis to describe regression models in general, for more information about estimating the model, testing for significance of parameters and construction of confidence intervals see for example [Hosmer and Lemeshow, 2000].

Stepwise logistic regression [Hosmer and Lemeshow, 2000] We will now describe the stepwise logistic regression algorithm, since it is widely used for building the logistic regression models.

Step(0) Assume that we have p independent variables. In Step(0), we first fit the intercept only model and compute its log-likelihood L_0. Then we try to add each of the p variables to this model and compute respective log-likelihoods of these models. Let us denote $L_j^{(0)}$ the log-likelihood of the model with variable x_j included. Then we perform the likelihood-ratio test for each model containing x_j versus the intercept only model, in particular we evaluate $G_j^{(0)} = -2(L_0 - L_j^{(0)})$ for each j. This quantities have $\chi^2(k-1)$ distribution, where k is number of categories of particular variable. We compute the p-values and choose variable with the lowest p-value ("the most significant one"). This is a candidate to proceed to the following step. But there is no guarantee, that this variable is statistically significant. So we choose level p_E to judge importance of variables. The model including variable with minimal p-value enters Step(1), if this p-value is smaller than p_E. If not, the algorithm stops here.

Step(1) We add respectively each of the remaining $p-1$ variables to the model obtained in previous step. Then we perform the likelihood-ratio tests, as in the previous step (the model with 2 variables against the one obtained in Step(0)) and choose the variable with minimal p-value. Then we again compare it to the p_E and if it is smaller, then we take the model with the variable, otherwise the algorithm stops.

Step(2) This step includes the check for backward elimination. It is possible, that when we added the variable in previous step, one of those added in the steps before the previous one is no longer important. So we try to remove the previous added variables one by one and again perform the likelihood ratio tests against the full model (the very last one). Now we find the maximal p-value and compare it to a pre-chosen value p_R. If it is bigger, we remove the corresponding variable from the model.

Step(S) Repeat these steps until all p variables are included in the model, or until we are not able to add or remove any variable to the model according to the rules.

This is the approach standardly mentioned in literature. In this thesis, we will use approach based not on p-values, but on AIC. The main idea is the same, but instead of checking the p-values, we choose from all possible models the one which is minimizing the AIC. Definition of AIC follows.

Definition 6. *The Akakaike Information Criterion (AIC) is defined as*

$$AIC = 2k - 2log(L)$$

where k is number of estimated parameters and L is the maximal value of likelihood function of the model.

As we can easily see from the definition, the AIC rewards goodness of fit through the likelihood function, but also penalizes high number of estimated parameters. This measure is good just for comparing models on the same dataset, the value is relative. And so the value itself does not tell us too much.

Whereas the approach based on p-values only considers significance of variables and their impact in the model, the approach based on AIC compares somehow the overall quality of the models. Also, because AIC penalizes high number of estimated parameters, it helps prevent overfitting.

1.2.3 Decision trees

Key terms Before we start, let us recall some basic definitions necessary to understand the problematic.

Definition 7. *[Safavian and Landgrebe, 1990]*

- *A graph $G(V, E)$ consists of a non-empty set of nodes V and edges E. If the edges are ordered pairs (v, w) of vertices, then the graph is said to be directed.*
- *A path in a graph is a sequence of edges of the form $(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)$. We say the path is from v_1 to v_n and is of length n.*
- *A directed graph with no cycles is called directed acyclic graph.*

Now we can continue with the definition of a tree.

Definition 8. *[Safavian and Landgrebe, 1990] A directed tree is a directed acyclic graph satisfying the following properties:*

1) There is exactly one node, called root, which no edges enter.

2) Every node except the root has exactly one entering edge.

3) There is a unique path from the root to each edge.

Definition 9. *[Safavian and Landgrebe, 1990]*

- *If (v, w) is an edge in a tree, then v is called the father of w and w is the son of v. If there is a path from v to w ($v \neq w$) then v is a proper ancestor of w and w is a proper descendant of v.*
- *A node with no proper descendant is called a leaf. All other nodes (except the root) are called internal nodes.*

Let us now introduce some special types of trees.

Definition 10. *[Safavian and Landgrebe, 1990] An ordered tree is a tree in which the sons of each node are ordered, normally from left to right.*

Definition 11. *[Safavian and Landgrebe, 1990] A binary tree is an ordered tree such that*

1) each son of a node is distinguished either as a left son or as a right son

2) no node has more than one left son nor more than one right son.

Design of the decision tree classifier - the basics. While designing the decision tree classifier structure, we need to classify correctly as much of the training sample as possible and generalize beyond the training sample. We also need the structure to be easy to update and to have as simple structure as possible. The task can be decomposed into the following parts:

1) The appropriate choice of tree structure.

2) The choice of feature subsets to be used at each internal node.

3) The choice of decision rule or strategy to be used at each internal node.

Let us denote the overall probability of error by P_e, a specific choice of tree structure by T, feature subsets to be used at the internal nodes by F and decision rules to be used at the internal nodes by d. Then we can solve our task by solving the optimization problem

$$\min_{T,F,d} P_e(T, F, d)$$

subject to limited training sample size.

This problem can be solved in two steps. In the first step, we minimize the overall probability error with respect to d, assuming T and F fixed. After we do that, in the next step, we minimize the error with respect to T and F with d fixed to the value obtained in the first step.

Information theory Before we have a look at particular algorithms used for growing the decision trees, let us introduce some basic definitions from information theory.

Definition 12. *[Siddigi, 2006] Information value (total strength) of characteristic X with levels $1, \ldots, n$ is defined as*

$$\sum_{i=1}^{n}((1-\pi_i)-\pi_i)\cdot log\left(\frac{(1-\pi_i)}{\pi_i}\right)$$

where π_i is a probability that client having the property i will be bad.

Note that information value is one of important indicators of variable predictevness.

Definition 13. *[Fyfe, 2000] Let $\mathbf{y} = (y_1, \ldots, y_N)^\top$ be vector of events. Then we define the entropy of $\mathbf{y}$ to be*

$$Entropy(\mathbf{y}) = -\sum_{i=1}^{N} p_i \cdot log\,(p_i)$$

By conditional entropy we understand the quantity

$$Entropy(i|\mathbf{y}) = p_i \cdot log\,(p_i)$$

where p_i is the probability that event i occurs.

Definition 14. *[Körting, 2018] Let $\mathbf{y} = (y_1, \ldots, y_N)^\top$. The gain for each i is defined as follows:*

$$Gain(\mathbf{y}, i) = Entropy(\mathbf{y} - Entropy(i|\mathbf{y}))$$

ID3 and C4.5 algorithm Let us now describe, how the ID3 algorithm works. To do that we will adopt the pseudocode from [Hssina et al., 2014]:

Inputs: *R: a set of non-target attributes, C: the target attribute, S: training data.*
Output: *returns a decision tree*
Start
Initialize to empty tree;
If *S is empty* ***then***
Return *a single node failure value*
End If
If *S is made only for the values of the same target* ***then***
Return *a single node of this value*
End If
If *R is empty* ***then***
Return *a single node with value as the most common value of the target attribute values found in S*
End If
$D \leftarrow$ *the attribute that has the largest Gain(D, S) among all the attributes of R*
$\{d_j, j = 1, \ldots, m\} \leftarrow$ *Attribute values of D*
$\{S_j, j = 1, \ldots, m\} \leftarrow$ *The subsets of S respectively constituted of* d_j *records attribute value D*
Return *a tree whose root is D and the arcs are labeled by* $d_1, \ldots, d_m$ *and going to sub-trees*
$ID3(R - \{D\}, C, S_1), \ldots, ID3(R - \{D\}, C, S_m)$
End

This algorithm has several disadvantages, for example that it is too sensitive to features with large number of values. This problem can be solved via using the C4.5 algorithm, which is the extension of ID3. C4.5 has two major properties, that the ID3 doesn't have:

- Can treat also continuous data.
- Is not so sensitive to features with large number of values.

Let us first focus on the first point - dealing with attributes with continuous ranges. Let us assume, that attribute C_i has continuous range. In the training set, we have got values $A_1, \ldots, A_n$ of this attribute. For each $j, j = 1, \ldots, n$ we partition the records into those having the value of C_i lower or equal to A_j and those having the value of C_i greater than A_j. Then we compute the *Gain* for each of those partitions and choose the one that maximizes it.

To achieve the second property, we use pruning. The point of pruning is that some subtrees which are "not needed" are replaced by a single leaf. As a consequence of this, the tree we obtain is not that large and complex and hence it is not too sensitive to features with large number of values. The replacement takes place, if the expected error rate in the subtree is higher than in the single leaf.

CART algorithm The main idea is the same as for the previously described algorithms. The main differences are:

- The resulting tree is always a binary tree.
- The splitting rule is based on Gini index (classification) or sum of squared errors (regression).
- Can handle missing values.

Definition 15. *[Loh, 2014] Let us have feature X with possible classes encoded by $1, \dots, j, \dots, J$.Let $p(j|t)$ be a proportion of class j learning samples in node t. Then the Gini index is defined as*

$$g(t) = 1 - \sum_j p^2(j|t)$$

The splits are chosen in such a way that the entity $\Delta i(s,t) = i(t) - p_L \cdot i(t_L) - p_R \cdot i(t_R)$ is maximized, where $i(t)$ is Gini index (classification) or sum of squared errors (regression), p_L is probability of being in the left branch, p_R is the probability of being in the right branch and t_L, t_R is the left and right subnode, respectively. Then a set of possible binary splits is constructed and the best split is chosen.

The missing values are handled by using surrogate splits.

In order to find a value of terminal node, let $C(i|j)$ be the cost of misclassifying a class j as class i. Then assign terminal node t to class $j^\star$, if it minimizes the missclasification cost

$$\sum_j C(j^\star|j)p(j|t) = \min_i \sum_j C(i|j)p(j|t)$$

Bagging and Random Forests (RF) Bagging is short for bootstrap aggregating. The goal is to reduce the variance. To do that, we would like to take many training sets, build separate models on those sets and than average the predictions to obtain one overall prediction. The problem is, that we usually do not have enough data to do this. To solve this problem, we use the bootstrapping approach, which basically means random sampling with replacement. So we generate B bootsrapped training data sets from the one we have. Then we build the decision tree for each of them and we obtain B predictions. Then we obtain the overall prediction by averaging them.

This approach has two major disadvantages. The first one is that we need to build the fully grown tree B times, so it can be computationally demanding. The second is, that the trees we obtain are highly correlated.

To obtain decorrelated set of decision trees, we apply the following improvement. For designing each tree we only use random subset of predictors, not all of them. What we obtain is a random forest.

Definition 16. *[Breiman, 2001] A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(\mathbf{x}, \Theta_k), k = 1, \dots\}$ where the $\{\Theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input vector $\mathbf{x}$.*

We will now introduce a theorem that says, that with increasing number of trees we always get better result, and hence we do not have to worry about overfitting caused by large number of trees.

Notation. Let us denote $h(\mathbf{x}, \Theta_k)$ shortly by $h_k(\mathbf{x})$.

Definition 17. *[Breiman, 2001]*

Let $h_1(\mathbf{x}), \ldots, h_K(\mathbf{x})$ be a set of classifiers. Let Y be a response and $\mathbf{X}$ an input vector, $(Y, \mathbf{X})$ is a random vector sampled from the training data. Then we define the margin function as

$$mg(\mathbf{X}, Y) = \frac{1}{K} \sum_{k=1}^{K} \mathbb{I}(h_k(\mathbf{X}) = Y) - \max_{j \neq Y} \frac{1}{K} \sum_{k=1}^{K} \mathbb{I}(h_k(\mathbf{X}) = j)$$

Definition 18. *[Breiman, 2001] The generalization error is defined as*

$$PE^{\star} = \mathbb{P}_{\mathbf{X},Y}(mg(\mathbf{X}, Y)) < 0)$$

The following theorem follows from the Strong Law of Large Numbers.

Theorem 1. *[Breiman, 2001] As the number of trees increases (i.e. for $K \to \infty$), for almost surely all sequences $\Theta_1, \Theta_2, \ldots$ $PE^{\star}$ converges to*

$$\mathbb{P}_{\mathbf{X},Y}(\mathbb{P}_{\Theta}(h(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} \mathbb{P}_{\Theta}(h(\mathbf{X}, \Theta) = j) < 0)$$

Proof. Can be found in [Breiman, 2001]. □

Variables importance [Friedman, 2001] In decision trees models, we are able to evaluate the so called variable importance, which is the relative influence of each variable on the model. In a single tree T, we obtain this quantity as follows:

$$\hat{I}_j^2(T) = \sum_{t=1}^{J-1} \hat{i}_t^2 \mathbb{I}(v_t = j)$$

where v_t is splitting variable associated with node t and $\hat{i}_t^2$ is corresponding empirical improvement in squared-error as a result of the split:

$$\hat{i}_t^2 = \frac{w_l \cdot w_r}{w_l + w_r} (\bar{y}_l - \bar{y}_r)^2$$

where w_l, w_r are sums of weights and $\bar{y}_l, \bar{y}_r$ are means of daughter responses.

Then, in a collection of trees $\{T_m\}_1^M$ we have

$$\hat{I}_j^2 = \frac{1}{M} \sum_{m=1}^{M} \hat{I}_j^2(T_m)$$

Variable importances are usually scaled into $[0, 1]$ interval, so that they are comparable.

1.2.4 Neural networks (NN)

Key terms Let us first summarize key terms from the theory of neural networks, since it is crucial to have them on mind in order to understand the problem.

1. Artificial neuron. Artificial neuron is a basic unit of artificial neural network. It consists of synapses and the body. Information is coming through input synapses, where is weighted, to the body, where is processed and then the processed information goes out through an output synapse.

2. Activation function. Activation function is a function of weighted sum of inputs that is performed in the body of neuron.

3. Neural network. Neural network is a network of neurons. It usually consists of input layer, one or more hidden layers and output layer. These are connected via synapses.

To formalize the first two points, let us denote $\mathbf{x}$ the vector of input data, $\mathbf{w}$ the vector of weights, y the neuron output and f the activation function. Then the following holds:

$$y = f\left(\mathbf{w}^T\mathbf{x}\right) = f\left(\sum_i w_i x_i\right), \;\; i = 1, \ldots, N \tag{1.5}$$

where N is number of input synapses. [Fyfe, 2000]

The weights assigned to the input synapses are adjusted via learning. There are more ways of learning, but we will focus on backpropagation, which is usually used for neural networks used in credit scoring.

Backpropagation algorithm Let us now describe, how the backpropagation algorithm works [Fyfe, 2000]. It is gradient descent, which means, that steps proportional to the negative of gradient of the function at the current point are taken in order to find a local minimum. The function to minimize here is the error function.

1. Initialization of weights. As first step, we need to initialize the weights to small random numbers.

2. Input pattern. We choose an input pattern $\mathbf{x}$ and apply it to the input layer.

3. Propagate forward. We propagate the activation forward till it reaches the output neurons.

4. Calculate δs for the output layer. Calculate δs according to the formula

$$\delta_i^o = (t_i - o_i) f'(Act_i), \;\; i = 1, \ldots, N \tag{1.6}$$

where t_i are the target values, o_i are the outputs, $f'()$ is the derivative of activation function, Act_i is the activation (weighted sum of neuron inputs) and N is number of output neurons.

5. Calculate δs for the hidden layer. Calculate δs according to the formula

$$\delta_i^h = \sum_{j=1}^{M} \delta_j w_{ji} f'(Act_i), \;\; i = 1, \ldots, N \tag{1.7}$$

where N is number of neurons in hidden layer, M is number of neurons in output layer and w_{ji} are the corresponding weights between hidden and output layer.

6. Update weights. Update all weights according to the formula

$$\Delta w_{ij} = \gamma \delta_i o_j \tag{1.8}$$

where γ is a learning rate which is a parameter and needs to be set in the beginning.

7. Repeat for all patterns. Repeat steps 2 - 5 for all input patterns.

Note, that we need the activation function to be differentiable in order to do this. In the algorithm above we use the mean square error as an error function (the algorithm is designed to minimize the mean square error). We can of course use different error functions.

Adaptive learning rate approach (ADADELTA) [Zeiler, 2012] In gradient descent, we have to select a constant global learning rate by hand, which is often rather art than science to do it correctly and effectively, eventually it is necessary to tune too many parameters, if we use for instance the momentum method, which is a common and simple extension of gradient descent (for more details please see [Zeiler, 2012]). This is an issue which leads to development of adaptive learning rate methods, where the learning rate is changing over time and the aim is that it is as optimal as possible at all iterations of the algorithm and possibly not too sensitive to initial parameters selection.

ADAGRAD The base of ADADELTA approach is so called ADAGRAD, which is a direct extension to gradient descent.

The update rule here is

$$\Delta w_{ij}^t = -\frac{\gamma}{\sqrt{\sum_{\tau=1}^{t} {g_\tau}^2}} g_t$$

where γ is a global learning rate and g_t is evaluated gradient of error function $\frac{\partial E}{\partial w_{ij}}$ at t-th iteration.

Note that the learning rate grows with the inverse of the gradient magnitudes, that means we have smaller rates for large gradients and larger rates for small gradients, which means, that the progress evens out over time.

This is obviously sensitive to choice of γ, as well as to initial choice of weights. The ADADELTA algorithm is derived from ADAGRAD and overcomes this sensitivity.

Use of Second Order Information Except from ADAGRAD, we also need to know how can be second order information used while dealing with our problem. Second order methods make use of the Hessian matrix in order to optimize the objective. Since computing the whole Hessian matrix would be expensive, only a diagonal approximation $diag(H)$ is used. Then the update rule becomes

$$\Delta w_{ij}^t = \frac{1}{|diag(H)| + \mu} g_t$$

where μ is a small constant to improve the conditioning. For more details please see for instance [Zeiler, 2012].

ADADELTA - motivation The ADADELTA method is derived from ADAGRAD, as already mentioned. The aim is to improve the main two drawbacks of ADAGRAD, which are fistly continual decay of learning rates during training, since we have the sum of squared gradients within the whole training history in the denominator, which is obviously increasing with increasing number of iterations and so the learning rate decreases, and secondly the need for a manually selected global learning rate. There are two ideas refering to the two drawbacks respectively introduced below. ADADELTA is derived based on these ideas.

ADADELTA - Idea 1 The first idea is, that instead of accumulating the sum of squared gradients over all time, we just take a time window of fixed length ω. By doing this, we make sure, that we only use the most recent gradients and the progress continues even after many iterations.

In order not to store ω past gradients, which is inefficient, the accumulation is implemented as an exponentially decaying average of the squared gradients. Assume at time t (at t-th iteration) this running average is $E[g^2]_t$ then we have:

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1 - \rho) g_t^2 \tag{1.9}$$

where ρ is a constant which needs to be set in the beginning. We require square root of these quantities, so we define

$$RMS[g]_t = \sqrt{E[g^2]_t + \epsilon} \tag{1.10}$$

where ϵ is a constant added to better condition the denominator. We obtain the following update rule:

$$\Delta w_{ij}^t = -\frac{\gamma}{RMS[g]_t} g_t$$

ADADELTA - Idea 2 The second idea is, that the units of weights and the weight updates should match. In ADAGRAD, this is not true, since the updates are unitless. It holds

$$units\ \ of\ \ \Delta w \propto units\ \ of\ \ g \propto \frac{\partial E}{\partial w} \propto \frac{1}{units\ \ of\ \ w}$$

$$\Delta w \propto \frac{\frac{\partial E}{\partial w}}{\frac{\partial^2 E}{\partial w^2}} \propto units\ \ of\ \ w$$

By rearanging the second order information method, we obtain

$$\Delta w = \frac{\frac{\partial E}{\partial w}}{\frac{\partial^2 E}{\partial w^2}} \Rightarrow \frac{1}{\frac{\partial^2 E}{\partial w^2}} = \frac{\Delta w}{\frac{\partial E}{\partial w}}$$

and hence, by substituting γ with $E[\Delta w^2]_t$,

$$\Delta w_{ij}^t = -\frac{RMS[\Delta w_{ij}]_{t-1}}{RMS[g]_t} g_t$$

ADADELTA - Algorithm Let us now introduce the ADADELTA algorithm [Zeiler, 2012].

Before the algorithm starts, parameters ρ and ϵ are required, as well as initial weights.

1. Initialize accumulation variables $E[g^2]_0 = 0$, $E[\Delta w^2]_0 = 0$
2. For all values of t do:
   - Compute g_t
   - Accumulate $E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) g_t^2$
   - Compute $\Delta w_{ij}^t = -\frac{RMS[\Delta w_{ij}]_{t-1}}{RMS[g]_t} g_t$
   - Accumulate $E[\Delta w_{ij}^2]_t = \rho E[\Delta w_{ij}^2]_{t-1} + (1-\rho) {\Delta w_{ij}^t}^2$
   - Apply $w_{ij}^{t+1} = w_{ij}^t + \Delta w_{ij}^t$

So the influence of ADADELTA on "classical" backpropagation as described above is in points 4. 5. and 6. where the rule for weights update is settled.

Often used activation and error functions. Let us now mention some often used activation and error functions. As we already mentioned above, the most common error function is the mean square error, which is of the following form:

$$E = \frac{1}{2} \sum_i (t_i - o_i)^2$$

where t is the target and o is the actual output. Another option is for example hyperbolic sine function defined as

$$f(x) = \sinh(|x|)$$

Often used activation functions are sigmoid function

$$f(x) = \frac{1}{1 + e^{-2cx}},$$

hyperbolic tangent

$$f(x) = \tanh(cx)$$

gaussian symmetric

$$f(x) = e^{-c^2 x^2}$$

and rectifier

$$f(x) = \max(0, x) \tag{1.11}$$

The rectifier function is probably the most popular activation function. Its smooth approximation is the function $f(x) = \log(1+\exp(x))$ called softplus. The derivative of softplus is the logistic function $f'(x) = \frac{\exp(x)}{1+\exp(x)}$.

Let us remark, that the type of neural network described above is called Multi Layer Perceptron (MLP) and is the most commonly used one in credit scoring problems.

Now we will introduce very important theorem that says, that we are able to approximate any reasonable function with arbitrary accuracy using neural network with one hidden layer.

Just before we do it, let us recall some important properties and definitions that we will need.

Definition 19. *[Hecht-Nielsen, 1989] We say, that function $f : [0,1]^n \to \mathbb{R}^m$ belongs to L_2, if each of f's coordinate functions is square integrable on the unit cube.*

Lemma 2. *[Hecht-Nielsen, 1989] Given any square-integrable function $g : [0,1]^n \to \mathbb{R}^m$, the series*

$$F(g, \mathbf{x}, N) = \sum_{k_1=-N}^{N} \cdots \sum_{k_n=-N}^{N} c_{k_1 \ldots k_n} \exp\left(2\pi i \sum_{j=1}^{n} k_j x_j\right) = \sum_{\mathbf{k}} c_{\mathbf{k}} \exp(2\pi i \mathbf{k}\mathbf{x})$$

where

$$c_{k_1 \ldots k_n} = c_{\mathbf{k}} = \int_{[0,1]^n} g(x) \exp(-2\pi i \mathbf{k}\mathbf{x}) d\mathbf{x}$$

converges to g in sense that

$$\lim_{N \to \inf} \int_{[0,1]^n} |g(\mathbf{x}) - F(g, \mathbf{x}, N)|^2 d\mathbf{x} = 0$$

Theorem 3. *[Hecht-Nielsen, 1989] Given any $\epsilon > 0$ and any L_2 function $f : [0,1]^n \to \mathbb{R}^m$, there exists a three-layer backpropagation neural network that can approximate f to within ϵ mean square error accuracy.*

Proof. Let $\epsilon > 0$. We would like to approximate each of the coordinate functions f_l of f with ϵ accuracy. According to Lemma 2, given $\delta_1 > 0$, there exists positive integer N and coefficients $c_{l\mathbf{k}}$ such that

$$\int_{[0,1]^n} |f_l(\mathbf{x}) - \sum_{\mathbf{k}} c_{l\mathbf{k}} \exp(2\pi i \mathbf{k}\mathbf{x})|^2 d\mathbf{x} < \delta_1$$

As the first step, we show, that we are able to approximate each sine and cosine term required in the Fourier series with arbitrary absolute accuracy using a subset of three-layer neaural network. In particular, we use the input layer, a subset H of hidden neurons and l^{th} output neuron. So we are able to compute any sum of the following form:

$$\sum_{i \in H} v_{li} \cdot s\left(\sum_{j=0}^{n} w_{ij} x_j\right)$$

where w_{ij} and v_{li} are weights, $w_{0j} = 0\ \forall j$, x_0 is the bias input and s is a sigmoid activation function.

Now note, that the arguments of sine and cosine functions $u(\mathbf{k}, \mathbf{x}) = 2\pi i \mathbf{k}\mathbf{x}$ are of the form

$$\sum_{j=0}^{n} w_{ij} x_j$$

By adjusting w_{i0} by $-(\pi/2)$ we can change a sine into cosine, and so we can concern ourselves just with sines. So we need to show, that for any given $\delta_2 > 0$ we are able to find coefficients v_{li} and w_{ij} such that

$$|\sin(u(\mathbf{k}, \mathbf{x})) - \sum_{i \in H} v_{li} \cdot s \left(\sum_{j=0}^{n} w_{ij} x_j \right)| < \delta_2$$

for each $\mathbf{x} \in [0, 1]^n$.
Choose w_{ij} such that

$$\sum_{j=0}^{n} w_{ij} x_j = \beta_i (u(\mathbf{k}, \mathbf{x}) - \alpha_i)$$

where β_i and α_i are real constants. In order to calculate each $\sin(u(\mathbf{k}, \mathbf{x}))$ and $\cos(u(\mathbf{k}, \mathbf{x}))$ we use the subset of hidden units $H_{\mathbf{k}}$. So we can rewrite the inequality as

$$|\sin(u(\mathbf{k}, \mathbf{x})) - \sum_{i \in H_{\mathbf{k}}} v_{li} \cdot s \left(\beta_i (u(\mathbf{k}, \mathbf{x}) - \alpha_i) \right)| < \delta_2$$

Given sufficient number of hidden layer units, this inequality can be always satisfied. Let us denote

$$S(\alpha, \beta, \mathbf{v}_l, \mathbf{x}) = \sum_{i \in H_{\mathbf{k}}} v_{li} \cdot s \left(\beta_i (u - \alpha_i) \right)$$

Note that u ranges over some closed interval $[d, e]$ for $\mathbf{x} \in [0, 1]^n$, because $\mathbf{x} \to u$ is a continuous mapping. So we need to approximate $\sin(u)$ on the interval $d \leq u \leq e$ using S. We partition the interval as follows:

$$d = \alpha_{i_1} < \cdots < \alpha_{i_{last}} = e$$

where $i_{p+1} = i_p + 1$ and $\bigcup_{p=1}^{last} \{i_p\} = H_{\mathbf{k}}$.

Clearly, S has an approximate form of staircase, where the step lenghts are determined by differences $\alpha_{i_{p+1}} - \alpha_{i_p}$ and heights by coefficients v_{li_p}. By setting the sigmoid gains β_{i_p}, $p > 1$ high enough, this basic staircase form can be always achieved.

Given the above facts, no matter how small $\delta_2 > 0$ is chosen, the sum $S(\alpha, \beta, \mathbf{v}_l, \mathbf{x})$ can be constructed such that it always remains within the δ_2 error band around $\sin(u)$.

Since the output unit l receives the inputs from the hidden layer units of each of the subsets $H_{\mathbf{k}}$, we are able to obtain the approximate Fourier series for f_l by multiplying all of the sine and cosine coefficients by multiplying it by appropriate combinations of real and imaginary parts of $c_{l\mathbf{k}}$. Call these sine and

cosine multipliers $a(l, \mathbf{k})$ and $b(l, \mathbf{k})$ respectively and let $y_l(\mathbf{x})$ be the output signal of the output unit l. We obtain:

$$F(f_l, \mathbf{x}, N) - y'_l(\mathbf{x}) = \sum_{\mathbf{k}} a(l, \mathbf{k})[\sin(2\pi i \mathbf{k}\mathbf{x}) - S(\alpha, \beta, v_l, \mathbf{x})] + \\ + b(l, \mathbf{k})[\cos(2\pi i \mathbf{k}\mathbf{x}) - S'(\alpha, \beta, v_l, \mathbf{x})]$$

where S' is the sum used for the cosine term.

Hence we obtain

$$\int_{[0,1]^n} |f_l(\mathbf{x}) - y'_l(\mathbf{x})|^2 d\mathbf{x} = \int_{[0,1]^n} |f_l(\mathbf{x}) - F(f_l, \mathbf{x}, N) + F(f_l, \mathbf{x}, N) - y'_l(\mathbf{x})|^2 d\mathbf{x} \leq \\ \leq \int_{[0,1]^n} |f_l(\mathbf{x}) - F(f_l, \mathbf{x}, N)|^2 d\mathbf{x} + \int_{[0,1]^n} |F(f_l, \mathbf{x}, N) - y'_l(\mathbf{x})|^2 d\mathbf{x} < \\ < \delta_1 + \delta_2^2 \sum_{\mathbf{k}} (a^2(l, \mathbf{k}) + b^2(l, \mathbf{k}))$$

So, given $\epsilon > 0$, we can choose δ_1, δ_2 sufficiently small to have

$$\int_{[0,1]^n} |f_l(\mathbf{x}) - y'_l(\mathbf{x})|^2 d\mathbf{x} < \frac{\epsilon}{n}$$

and so

$$\int_{[0,1]^n} |f(\mathbf{x}) - y'(\mathbf{x})|^2 d\mathbf{x} < \epsilon$$

for L_2 functions. □

Dropout [Srivastava et al., 2014]

In large networks, there might be problem of overfitting. In order to deal with this problem, we will use the dropout approach. The point of this approach is, that during the training, in each iteration we ignore random subset of network nodes. We select each node with probability p and ignore with probability $1 - p$. Then, in the test phase, we use the network with all nodes, but we adjust the weights by multiplying them by p.

Variables importance Similarly as for decision trees, also in a neural network model we can compute variables importances. For purposes of this thesis it is done as proposed by [Gedeon, 1997]. We define a contribution of a hidden neuron to an output neuron as follows:

$$P_{jk} = \frac{|w_{jk}|}{\sum_r |w_{rk}|}$$

where w_{jk} is a weight between nodes j and k.

The contribution of an input neuron to an output neuron is then

$$Q_{ik} = \sum_r P_{ir} \cdot P_{rk}$$

1.3 Measures of model quality

Once we have a model, we need to judge somehow its quality. There are several ways how to do it. In this thesis, we will use Gini coefficient and accuracy, sensitivity and specificity, which are measures of misclassification. Definitions follow.

Definition 20. *[Kozmík, 2006] Gini coefficient of scoring function s is defined as*

$$Gini(s) = 2\int_0^1 (F^G(p) - F^B(p))dF^G(p)$$

where $F^G(p) = \mathbb{P}[s(\mathbf{X}) < p|good]$, $F^B(p) = \mathbb{P}[s(\mathbf{X}) < p|bad]$ and $\mathbf{X}$ is a random vector of characteristics of a client.

The Gini coefficient tells us, how good is our model compared to making decisions completely randomly. As we can easily see from Definition 20, the more different are distributions of goods and bads according to our model, the higher Gini coefficient. Perfectly discriminating model would have Gini coefficient equal to one.

Definition 21. *[Siddigi, 2006] Let B_{obs} be a set of observed bad cases, G_{obs} a set of observed good cases, $B_{correct}$ a set of correctly classified bad cases and $G_{correct}$ a set of correctly classified good cases. Then we define*

- ***Accuracy*** *is defined as a ratio of correctly classified cases (true positives and negatives) over total number of cases, i.e.*

$$accuracy = \frac{|B_{correct}| + |G_{correct}|}{|B_{obs}| + |G_{obs}|}$$

- ***Sensitivity*** *is defined as a ratio of correctly classified good cases over total number of observed good cases, i.e.*

$$sensitivity = \frac{|G_{correct}|}{|G_{obs}|}$$

- ***Specificity*** *is defined as a ratio of correctly classified bad cases over total number of observed bad cases, i.e.*

$$specificity = \frac{|B_{correct}|}{|B_{obs}|}$$

Note that specificity is probably much more crucial than sensitivity, because it gives us information about the amount of bad clients which were classified as good, in the other words what amount of bad clients would we accept. Sensitivity tells us what amount af good clients would we reject, which is also a thing we do not want to do, because if we reject too many good clients, we are losing money, but it does not increase the credit risk we have to carry. Accuracy gives us information about overall precision of our model.

2. Computational part

In this chapter, we would like to discuss some practical issues. This section can be understood as an extension of the theory with respect to our special case. We will define good and bad clients and discuss variables selection. We will have a look at motivation of binning variables and introduce selected approaches to variables binning. We make some comments on models validation and comparison and we have a look at problem of neural network architecture and parameters selection. Finally, we will discuss early stopping.

2.1 Definition of "bad"

The very first thing we need to do before we start to develop any model is to define what means "bad", i.e. what is the response. In this thesis, we will use the following definition.

Default is defined as 90 days past due, that means, that the client stopped repaying 3 months ago and he still doesn't pay anything. We choose a 1 year long window, so for purposes of this approach we choose only clients with at least 15 months history (if the default started in 12th month, we observe it 3 months later). That means we consider defaults during the first year.

Another option of "bad" definition is for instance 60 days past due, which is an analogy of our definition, but is more strict, so we would observe more bad clients. Also use of various qualitative criteria is common, usually in combination with some "days past due" definition.

Being "good" is defined as not being "bad". Note that the "indeterminate" clients as mentioned in Section 1.1.1 are not in the dataset, because "indeterminate" are those who are neither "good" nor "bad" since we are not able to decide. But in our dataset, according to the previous paragraph, we only have clients for whom we are always able to decide, since only clients with at least 15 months history are selected.

2.2 Variables selection

An important part of model development is selection of variables. We need to choose the variables so that we do not use variables we do not need, but also not to lose information. Note that this step refers to Section 1.1.3.

Basic approach is to evaluate strength of relationship between each variable and response. To do this, we compute Gini coefficient (see Definition 20) of simple logistic model of form $response \sim variable$. Then we choose only variables with "sufficiently" high coefficient. The question is how to choose the treshold. In this thesis the treshold is set to 0.1, if used. It is because it is the highest possible value for which there is still reasonable number of variables left.

Another option is to use only variables with "sufficiently high" information value (see Definition 12). As a rule of thumb [Siddigi, 2006], it holds, that $IV < 0.02$ means unpredictive, $0.02 < IV < 0.1$ means weak, $0.1 < IV < 0.3$ means medium, $0.3 < IV < 0.5$ means strong and $IV > 0.5$ should be checked, is

suspicious. In this thesis the treshold is set to 0.1, if used. It is because for lower values the variable is classified as weak and at the same time it is the highest possible value for which there is still reasonable number of variables left.

Standard option is stepwise regression, which was already discussed in Section 1.2.2.

We will also try to combine these approaches.

2.3 Binning

Let us now discuss a problem of binning of continuous variables. To bin a variable means to categorize it, i.e. to change a continuous variable into a categorical by dividing it into distinct intervals according to some rule. There are many options how this can be done. We will introduce two of them in this section and then we will apply them. Results are presented in Chapter 3. As already mentioned in Section 1.1.3, there are reasons why to do it and why not to do it. It is in scope of this thesis to explore both cases.

The main reason why not to bin variables definitely is that we are losing information. If we have age of the client as a continuous variable with range from 18 to 70 and we make bins $[18, 30), [30, 50), [50, 70]$, it means that we say that from the point of view of our model 20 is the same as 27, 32 is the same as 46 etc. Which doesn't matter, if it really is the same from the perspective of the model, but if we don't do the binning carefully, it can easily happen, that this is not true. And then it is a problem and we are not just losing information, but even adding incorrect information to the model.

But there are also reasons why to bin the variables. The very important one is, that there is no guarantee, that the dependency of response on the variable will behave linearly. It does not necessarilly hold, that the higher age is the client the better (or worse) he will be. In these cases binning is appropriate if we want to use a linear model. And if it is done correctly, then it will probably lead to much better results. In our dataset, variable CLIENT INCOME is such an example. To ilustrate the behavior of this variable, we divided it into intervals of the same length and plotted good and bad rates in each group into Figure 2.1.

We will present results from models with both binned and not binned variables. We will do the binning automatically, using two different approaches, which we will describe now.

Recursive Partitioning (RP) Let us describe the main idea of Recursive Partitioning algorithm we will use for binning of continuous variables. Technically, we will do it in R software using function *smbinning*(). Generally, the algorithm can be described in the following 3 steps:

1. Assume that we have m covariates and the response. The first step is testing the global null hypothesis of independence of any of the m covariates and the response. If we are not able to reject this hypothesis, the algorithm stops. Otherwise select the covariate X_j with the strongest association with response.

2. Let $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_m$ be a sample space. Split $\mathcal{X}_j$ into two disjoint subsets according to criterion described below.

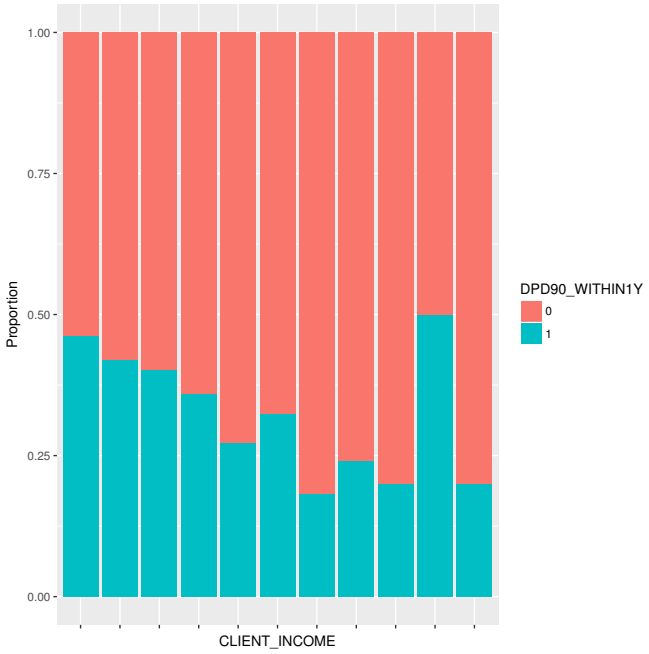

Figure 2.1: Good and bad rates - CLIENT INCOME binned into intervals of the same length

3. Recursively repeat steps 1. and 2.

So the approach is tree-based and the splitting criterion (step 2.) is based on the permutation test framework. The goodnes of a split in a chosen variable $X_{j^\star}$ is evaluated by the two-sample linear statistics of form

$$T_{j^\star}^{A}(\mathcal{L}_n) = \sum_{i=1}^{n} \mathbb{I}(X_{j^\star i} \in A) Y_i$$

where $\mathcal{L}_n$ is a learning sample, A is a subset of $\mathcal{X}_{j^\star} = A \cup (\mathcal{X}_{j^\star} \setminus A)$ and Y_i is the response for i-th observation.

This induces a two-sample statistic measuring the discrepancy between the samples $\{Y_i | X_{ji} \in A, i = 1, \ldots, n\}$ and $\{Y_i | X_{ji} \notin A, i = 1, \ldots, n\}$. The split where the standardized linear statistic is maximized is chosen.

For more details see [Hothorn et al., 2016].

WOE based binning This approach is based on weight of evidence and information theory. First of all, the numeric variable is splitted into n initial classes with similar frequencies. Then these classes are merged so that the classes with similar WOE are joined together. This is done as long as the information value of all groups is higher than a given treshold. Practically, we will do it in R software using function *woe.binning*(). Note that the number of initial classes used in this thesis was 20 and the stopping criterion was, that the resulting information value does not decrease more than 10% compared to the previous one.

2.4 Models validation and comparison

Once we have the model, we need to check if it behaves reasonably and we also need to compare the models somehow. There are several ways how to do it. The way we will use in this thesis is computing the Gini coefficient, accuracy, sensitivity and specificity, as already mentioned in Section 1.3. Since these indicators depend on the split of the data sample into train and validation, we will use k-fold cross validation approach if possible. In some cases there are too many models compared to do the cross validation - it would be computationally demanding. In cases where cross-validation is not performed, the models are compared always with respect to coefficients computed based on the same data split.

K-fold cross validation The idea of k-fold cross validation is, that we divide the data sample into k parts and we always take one of these parts as a validation sample and the rest as a train sample. We train the model k times and compute the indicators for all of these models. Than we can average the indicators and we have much better idea about the model performance.

2.5 Problem of neural network architecture and parameters selection

When we want to use neural networks, we need to select number of hidden layers, number of hidden neurons and the ADADELTA parameters ρ (see Equation 1.9) and ϵ (see Equation 1.10). There are some rules of thumb, but they differ a lot, so this is not very reliable. So we will use these rules as a hint, but we will not follow them strictly. We will need to try more values to decide what will be the best. Rules of thumb regarding number of hidden neurons we will use are [Basheer and Hajmeer, 2000]:

- Number of hidden nodes should be approximately equal to square root of number of input nodes (if we have one output node).
- Upper bound for number of hidden nodes is number of input nodes plus one.

Epochs Number of epochs is the number of times to iterate the dataset. In this thesis this is set to 10 where cross-validation is not done. If cross-validation enabled, the number of epochs is tuned automatically based on convergence behavior of the cross-validation models.

2.6 Stopping metric

Stopping metric is the metric to use for early stopping of neural networks and random forests training. In this thesis we use the logloss, which is defined as

$$logloss = -(y \log(p) + (1 - y) \log(1 - p))$$

where y is the real value of response (0 or 1) and p is the predicted probability of y being 1. The formula above is for one observation, for the whole dataset it is a simple average.

The training is stopped, when the improvement of logloss is lower than 0.001. In Figure 2.2 we can see the logloss for $y = 1$.

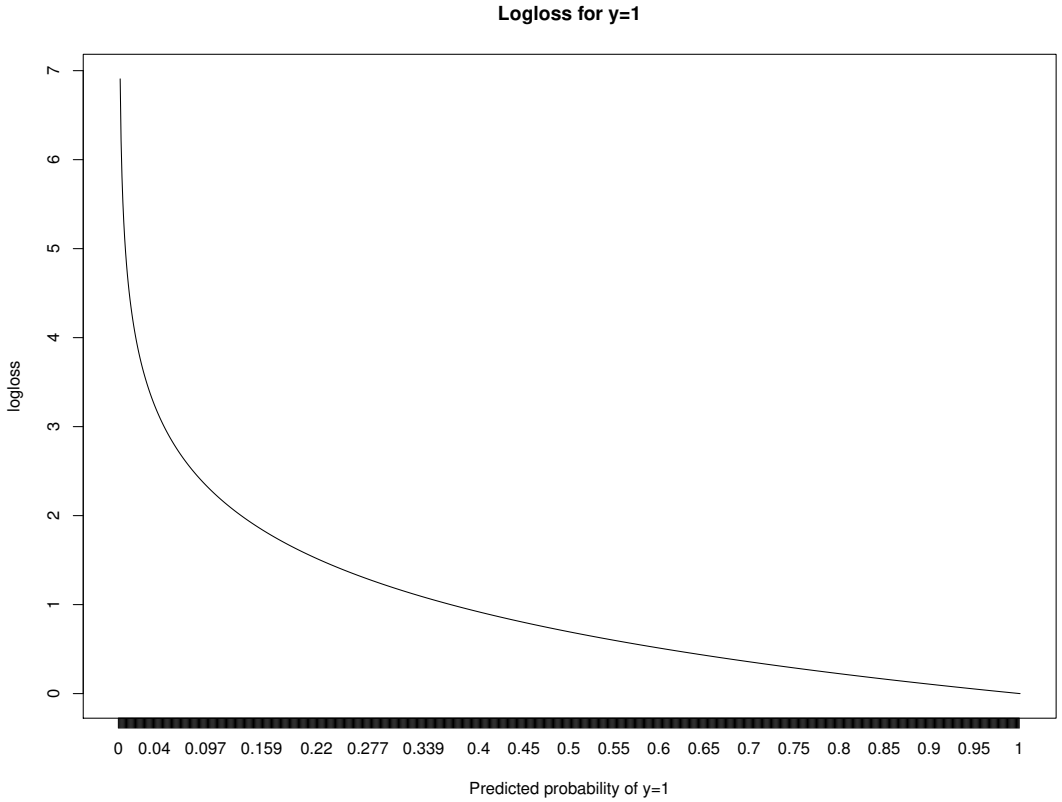

Figure 2.2: Logloss for y = 1.

3. Applied part

In this chapter we will introduce the dataset and then we will show and discuss results we obtained by using the methods described in previous chapters.

All models are implemented using R [R Development Core Team, 2008] and H2O [The H2O.ai team, 2015] via built-in functions. For details see attached R code (Attachment A.1). The data preprocessing was done using SQL Server Management Studio [Microsoft, 2017] (see Attachment A.2).

The objective (in the sense of Section 1.1.1, Point 1) is to minimize credit risk, i.e. decrease possible losses, by predicting defaults of clients based on the 90 DPD definition described in previous sections. We will compare different approaches. We compare two options of binning of numeric variables for logistic regression model approach. We also train a neural network model and a random forest model. Then we will compare these three approaches. Because we want most importantly to minimize credit risk, we want to classify correctly as many truly bad clients as possible - that is our priority.

3.1 Data

For further analysis, we will use dataset provided by PROFI CREDIT Czech, a.s. Since the dataset is confidential, we are not allowed to provide it and so it is not included in Attachments. We have got a portfolio of consumer loans. The data contains 267 678 observations, from which 40 487 can be used for the model development, since they are approved loans with at least 15 months history. From these 16 889 are defaults according to 90 DPD definition which we use. The rest are rejected loans or records not usable for the development, since their history is not long enough or the data is not reliable. The summary of dataset composition is shown in Table 3.1.
Note that missing values in categorical variables are treated as another level. Missing values in numeric variables are treated as zeros when the variable is used as continuous and as a separate bin, when the variable is binned.
From the dataset, we take variables which might be relevant for us. These are listed in List 3.1.

Type	Count	Perc. of dataset	Perc. of approved
Good	23 598	8.8%	58.3%
Bad	16 889	6.3%	41.7%
Reject	159 785	59.7%	-
Not usable	67 406	25.2%	-

Table 3.1: Summary of the dataset.

In Figure 3.1 we can see map of relationships of variables. For purpose of plotting, numeric variables were binned based on WOE. The interpretation of the

plot is, that the more correspond the variables (levels) to each other, the closer to each other they are. For each two levels number of matches was computed. Based on these numbers the correspondence is visualized. We can only see variables with highest total numbers of matches, i.e. those which are the most related to the others. The map was created using VOSviewer [Eck and Waltman, 2018], see the attached source files (Attachment A.3).
Note that DPD90_WITHIN1Y is the response, where 0 is good and 1 is bad. We can observe, that clients with no other costs, total income between 13 000 and 20 000, shorter agreed loan length (up to 30 years), unmarried and not living in shared household are more likely to be bad. On the other hand, clients with zero solus count, nonzero other costs, zero children costs, married, with own housing and with higher agreed loan length are more likely to be good.

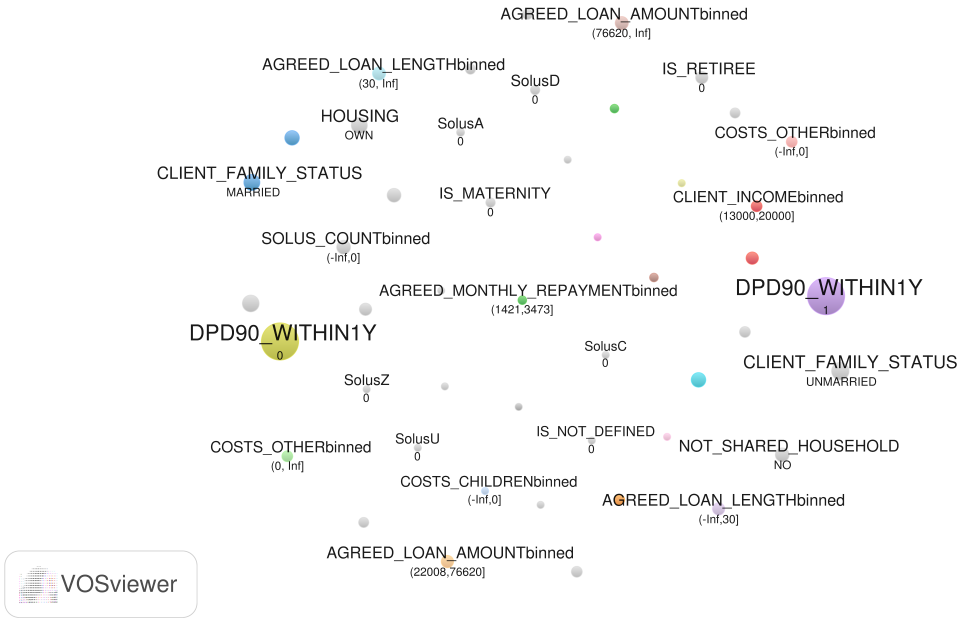

Figure 3.1: Variables connections.

In Figure 3.2 we can see distribution of good and bad clients according to categorical variables, where the difference is relevant. The variables where the default rates in all levels seem to be the same are not shown.
We can observe, that by unmarried clients there is higher chance to be bad, by married it is the other way around. There is higher chance to be good for clients with own housing. By clients with private lease it is the other way around. There is higher chance to be good for clients living in shared household. We can see that having SolusA, SolusB or SolusD sign will be probably relevant. In particular, SolusA means that client owes 2 repayments to a member of the Solus association. SolusB means that client owes 3 repayments and the meaning of SolusD is that the client owes 1 repayment.

Solus is an interest association of artificial entities which helps to prevent overindebting of consumers and to decrease potential financial loss of creditors. The members of the association share information about clients in Solus registers, which means that if one of the members have a negative experience with a client, the other members obtain this information through the register. The information

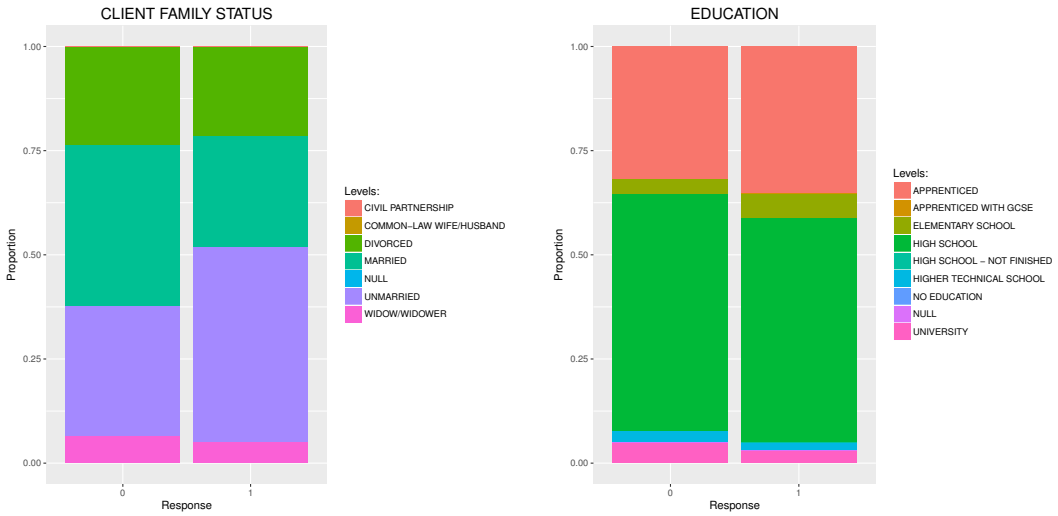

Figure 3.2: Distribution of categorical variables - part 1. Response 90 DPD, 0 is good, 1 is bad.

shared can be also positive, but all the information from this register obtained in our dataset is negative. To sum up, a sign from Solus register in our case generally means an external negative information.

In Figure 3.3 we can see distribution of those numeric variables, where there is obviously difference between good and bad clients. The other variables are not shown, since the plots for good and bad clients look very similar.

We can see, that it seems that bad clients usually have lower or no other costs, which is an interesting observation, because we would expect that clients with higher other costs will have less money left in total and hence will more likely get in trouble. Maybe it means that clients with higher other costs are more responsible, but this would be rather a psychological question. We can see, that good clients are generally of higher age. That seems logical, since we can expect that older clients will be more responsible and also will have probably more stable job. The NRKI result is higher for good clients, which is expected, since this is an external information from the NRKI register (NRKI stands for non-banking register of client information) and higher NRKI score means better client. We can see, that employment length of good clients is in general higher, which is again logical, because we can expect that the income of clients who are employed for a longer time is more stable.

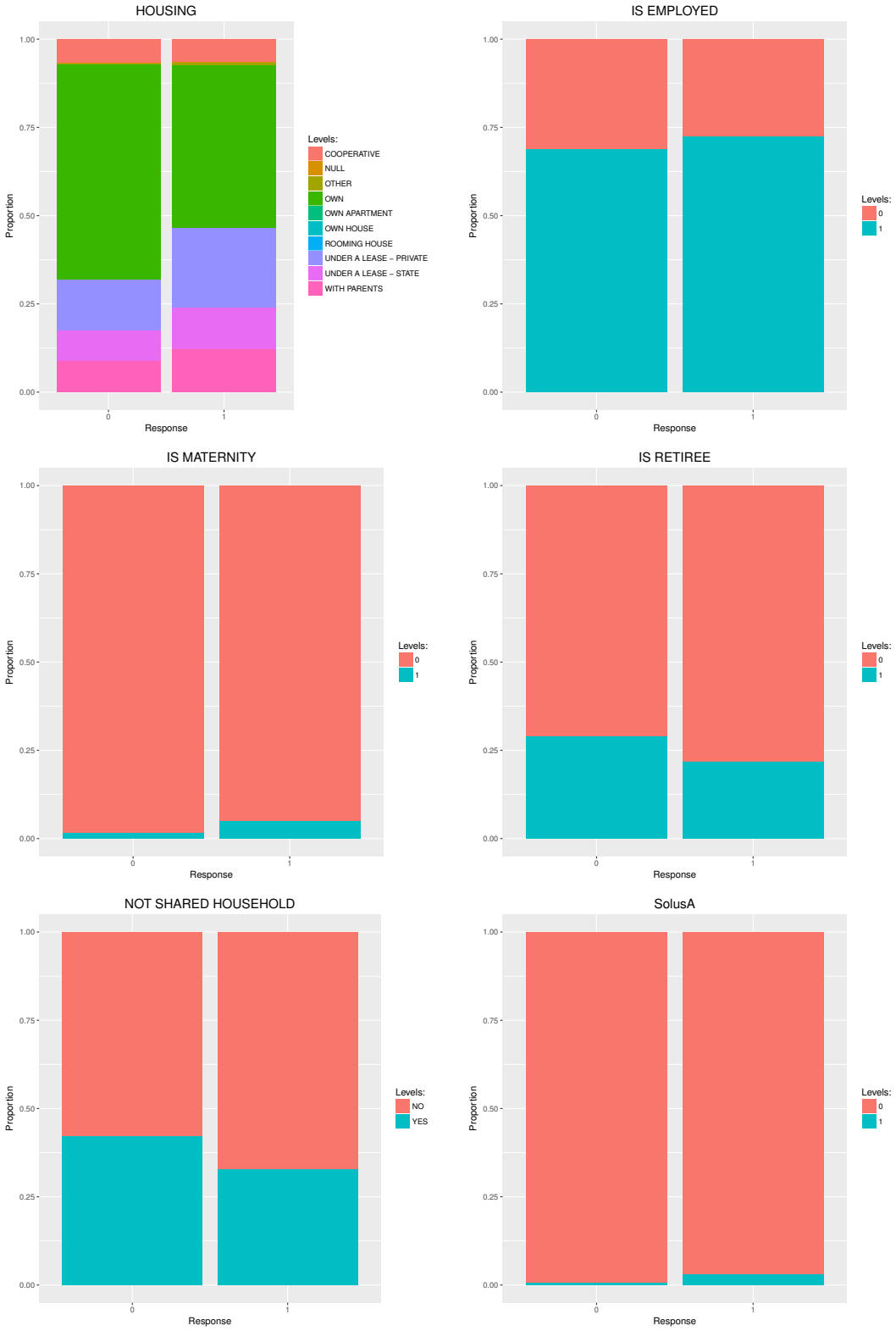

Figure 3.2: Distribution of categorical variables - part 2. Response 90 DPD, 0 is good, 1 is bad.

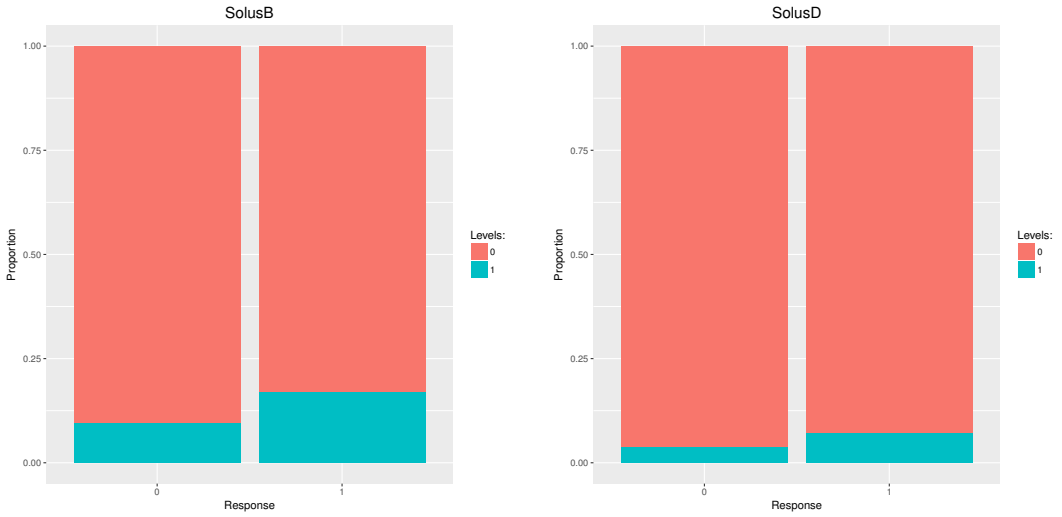

Figure 3.2: Distribution of categorical variables - part 3. Response 90 DPD, 0 is good, 1 is bad.

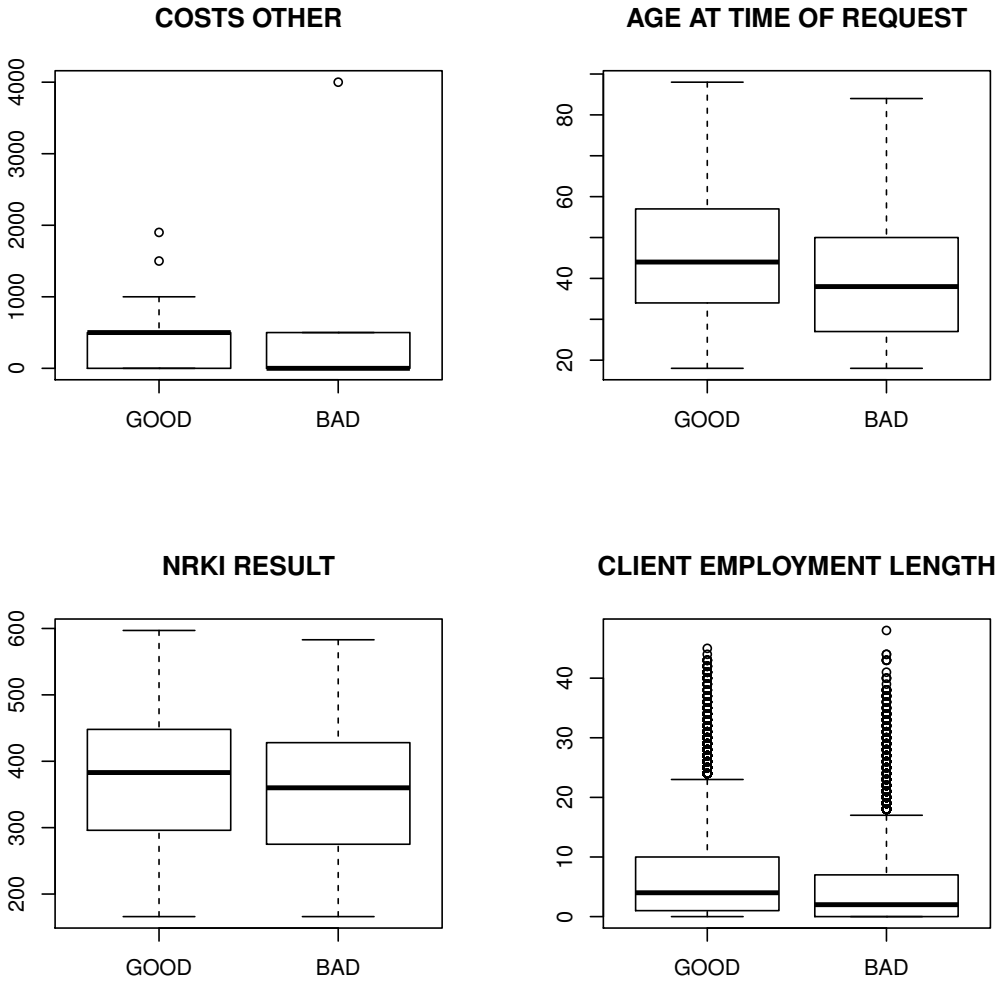

Figure 3.3: Distribution of numeric variables. Response 90 DPD, 0 is good, 1 is bad.

FREE_SOURCE_OVER_INSTALLMENT_RATIO: Ratio of total free sources and installment. Treated as numeric.

CLIENT_INCOME: Client income. Treated as numeric.

FREE_SOURCES: Clients free sources. Treated as numeric.

INCOME_OTHER: Other income. Treated as numeric.

INCOME_TOTAL: Total clients income. Treated as numeric.

CLIENT_PARTNER_COST: How much gives client to the partner. Treated as numeric.

COSTS_OTHER: Other costs. Treated as numeric.

COSTS_CHILDREN: Costs for children. Treated as numeric.

COSTS_TOTAL: Total costs. Treated as numeric.

AGREED_LOAN_LENGTH: Agreed length of the loan. Treated as numeric.

AGREED_LOAN_AMOUNT_WITH_INTEREST: Agreed loan amount enlarged by interest. Treated as numeric.

AGREED_LOAN_AMOUNT: Agreed loan amount. Treated as numeric.

IS_SURETY: Information about whether there is surety. 0 means no, 1 means yes. Treated as categorical.

AGREED_MONTHLY_REPAYMENT: Agreed monthly repayment. Treated as numeric.

IS_RETIREE: Information about whether the client is retiree. 0 means no, 1 means yes. Treated as categorical.

IS_MATERNITY: Information about whether the client is at maternity. 0 means no, 1 means yes. Treated as categorical.

IS_EMPLOYED: Information about whether the client is employed. 0 means no, 1 means yes. Treated as categorical.

IS_NOT_DEFINED: If there is 1 in this field and at the same time there is 1 in IS_EMPLOYED, then client is employee. Treated as categorical.

IS_FOREIGNER: Information about whether the client is foreigner. Treated as categorical.

AGE_AT_TIME_OF_REQUEST: Clients age at time of request. Treated as numeric.

SolusA - SolusZ: Information about whether client has particulat sign in Solus register. All this information is negative. Treated as categorical.

SOLUS_COUNT: Number of signs in Solus register. Since the information from the Solus register is always negative, we can expect, that worse client will have more signs. Treated as numeric.

SOLUS_QUERY: Information about whether there was a query regarding the Solus register from the side of PROFI CREDIT. Treated as categorical.

NRKI_RESULT: NRKI score of the client. Higher numbers correspond to better clients. Treated as numeric.

List 3.1: List of all variables - part 1

NRKI_CAT: Negative information from NRKI register. Treated as categorical.

CLIENT_FAMILY_STATUS: Client family status. Treated as categorical.

CLIENT_REGION_PERMANENT_ADRESS: Region of clients permanent adress. Treated as categorical.

CLIENT_EMPLOYMENT_LENGTH: Length of client employment. Treated as numeric.

REQUEST_IN_INSOLVENCY: Information about whether the client was in insolvency at time of request. 0 means no, 1 means yes. Treated as categorical.

REPAYMENTS_FROM_REGISTER: Other repayments, information from the Solus register. Treated as numeric.

BUILDING_SOCIETY_ACCOUNT_PENSION_INS: How much pays client on building society account and/or pension insurance. Treated as numeric.

OTHER_REPAYMENTS: Other repayments. Treated as numeric.

OTHER_REPAYMENTS_PROFICREDIT: How much pays client at other products of PROFI CREDIT.

CHILDREN_COUNT: Number of children. Treated as categorical (6 categories).

HOUSING: Clients housing. Treated as categorical.

EDUCATION: Clients education level. Treated as categorical.

NOT_SHARED_HOUSEHOLD: Information about whether the client lives in shared household. Treated as categorical.

List 3.1: List of all variables - part 2

3.2 Outputs

In this section we will present results of empirical experiments we did. First we compare the binning approaches themselves and then we present results from logistic regression models, neural networks models and random forests models.

3.2.1 Comparison of binning approaches

As already discussed in Section 2.3, we might need to bin the continuous variables in order to obtain better results. Therefore we will have a look at the two approaches described in Section 2.3 and first we will look at the differences in the binning itself, then we try to estimate some models using binned variables so that we can see the impact it will have on the model quality.

In this subsection, we will focus on results of the two different binning approaches. In Figure 3.4, we can see the most significant differences between the two approaches. We can see, that most of the variables behave in the same way w.r.t. weight of evidence, the difference is just in number of categories (and of course splitting points). The only exception is agreed loan amount, where the RP based binned variable behaves in a slightly illogical way. Let us remark, that for variables *free source over installment ratio*, *client partner cost* and *building society account pension ins* it was not possible to find any significant splits during RP binning. Generally, we can observe, that RP binning algorithm tends to make more new levels. This is reasonable, since the WOE approach begins with a fixed number of bins and then just merges them according to their similarity, whereas the RP approach is a bit more sophisticated and goes the other way around, it starts with the whole range and splits it according to a splitting rule. One could say, that having too many levels will increase the model complexity, but here this will not be a problem, since the number of levels is not that high. The potentional problem here is the low number of levels coming from the WOE approach, since there is a high risk of loosing important information, because the number of levels for some variables is really low. But it does not mean that it really is a problem, we will see in Section 3.2.2 what is the impact on model quality.
Note that in the plots there are not included the missing values bins. It is because the missing values are not subject of the binning, they are just simply taken as an extra category and hence it would be rather confusing to show these bins here in this context.

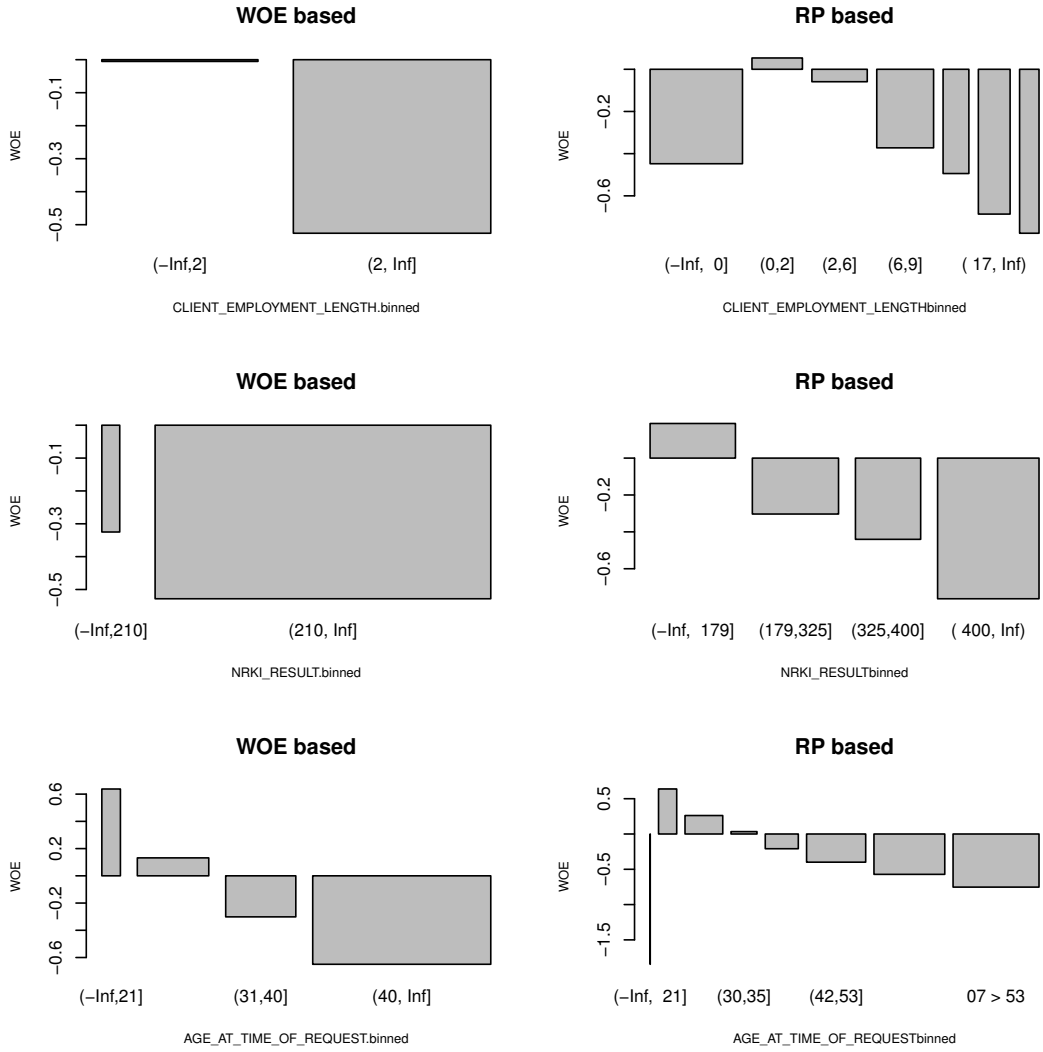

Figure 3.4: Comparison of binning approaches - part 1

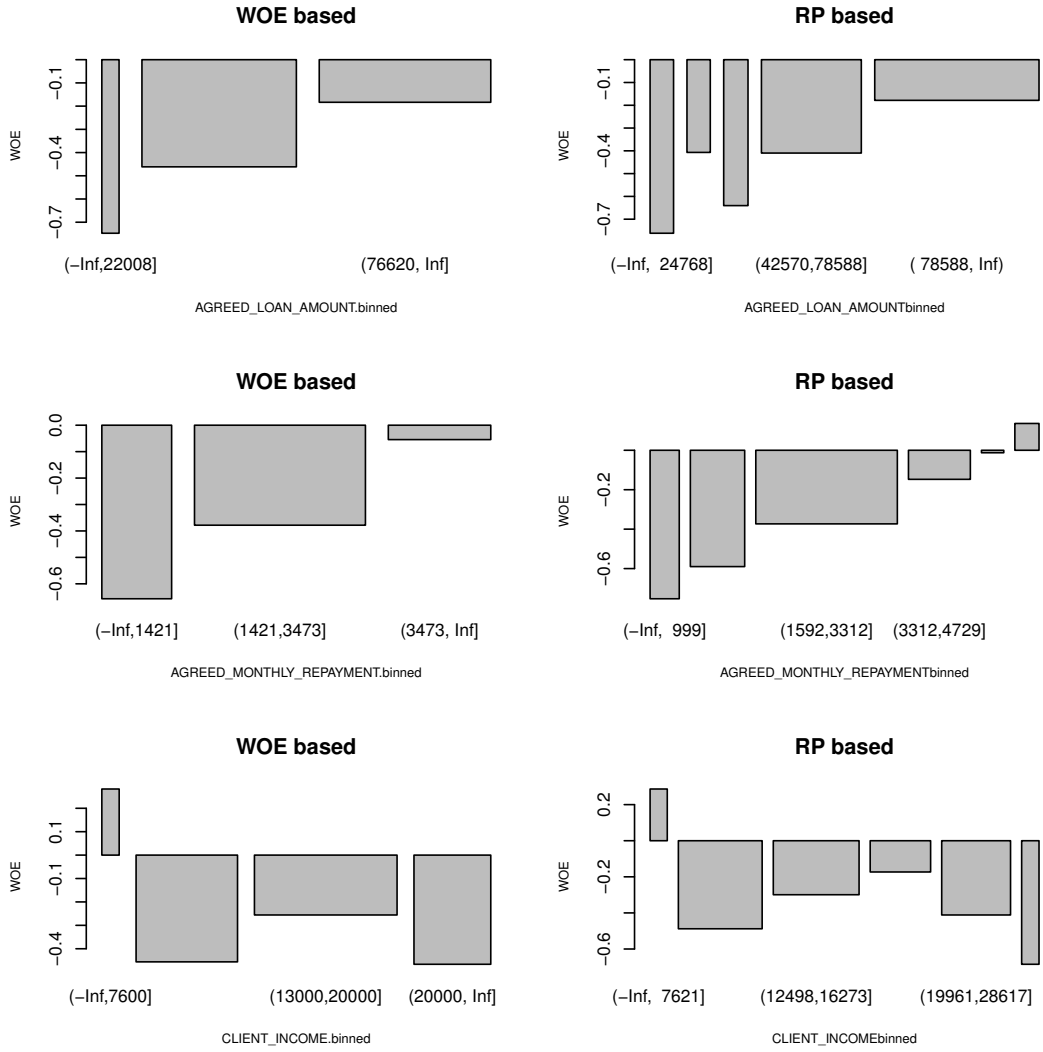

Figure 3.4: Comparison of binning approaches - part 2

3.2.2 Logistic regression results

Let us now have a look at results of logistic regression models based on theory introduced in Section 1.2.2. We used two different methods of binning variables and we also tried not to bin them at all. We tried combinations of three variables selection methods - setting Gini treshold, IV treshold and stepwise regression, as mentioned in Section 2.2. All mentioned approaches were described in previous sections. The quality of all fitted models with respect to Gini coefficient is shown in Table 3.2.

We can see, that we reached the highest Gini coefficient using stepwise regression with all and not binned variables. Combination of recursive partitioning binning and stepwise regression also gives us a good result. Since cross-validation was not performed here, the comparison is rather approximate and so we will focus on all the six models having Gini coefficient over 0.38. The results are not perfect, since the Gini coefficient is a number between 0 and 1 and the higher, the better. But it is expected, as we will discuss a bit more in Section 3.2.5. We can observe, that in our particular case, removing variables from the model using any other rule than stepwise regression decreases the model quality. On the other hand, removing variables using stepwise neither decreases nor increases the quality. We can observe, that binning the variables almost always decreased the model quality, the only exeption is the case where only Gini treshold was used. Here we obtain the best results for models with WOE binning (both with and without stepwise, the difference is negligible).

We perform 5-fold cross-validation for the six selected models. Obtained characteristics can be seen in Table 3.3. We can see, that according to both Gini coefficient and accuracy we would prefer model with not binned variables and no tresholds. We can see, that the model with use of stepwise regression and without gives almost the same result, so we prefer the one without stepwise, because it is computationally much less demanding (fitting the model without stepwise takes 3.3 seconds and fitting the model using stepwise takes 1285.2 seconds) and the impact on result is minimal. Note that this means, that we keep all the variables in the model, since removing variables always leads to worse result. From the common sense point of view it does make sense, because in the very beginning only variables which seem relevant and logical to include were considered. Since we have large amount of observations, there is no danger of overfitting caused by large number of variables. So we can keep all the variables without getting in trouble.

We compute also specificity and sensitivity on validation data for the best model. We obtain that sensitivity is 0.9371, which is quite good, but sensitivity is not the most important thing, because it measures misclassification of truly good clients, which is not that crucial, as discussed in Section 1.3. Much more crucial is specificity, which measures misclassification of truly bad clients and equals 0.2289, which is not very good.

In Table 3.4 we can see selected 5 most significant variables from the best model with estimated parameters.

The very first thing we can notice is, that all the variables which we have in Table 3.4 are shown in Section 3.1 as there are differences in distribution of good and bad clients and hence it is expected, that these variables will be important. We can see, that the parameter for SolusA YES, which means for clients having

Binning	Gini tr.	IV tr.	Stepwise	GINI coef.
NO	NO	NO	NO	**0.3950**
NO	NO	NO	YES	**0.3955**
WOE num.	NO	NO	NO	0.3792
WOE num.	NO	NO	YES	0.3784
RP	NO	NO	NO	**0.3871**
RP	NO	NO	YES	**0.3880**
NO	0.1	NO	NO	0.3456
NO	0.1	NO	YES	0.3456
WOE num.	0.1	NO	NO	0.3639
WOE num.	0.1	NO	YES	0.3640
RP	0.1	NO	NO	0.3501
RP	0.1	NO	YES	0.3501
NO	NO	0.1	NO	**0.3868**
NO	NO	0.1	YES	**0.3863**
WOE num.	NO	0.1	NO	0.3107
WOE num.	NO	0.1	YES	0.3107
RP	NO	0.1	NO	0.3294
RP	NO	0.1	YES	0.3288
NO	0.1	0.1	NO	0.3455
NO	0.1	0.1	YES	0.3456
WOE num.	0.1	0.1	NO	0.3107
WOE num.	0.1	0.1	YES	0.3107
RP	0.1	0.1	NO	0.3273
RP	0.1	0.1	YES	0.3267

Table 3.2: Comparison of various binning approaches and indeces treshold settings with respect to Gini coefficient. Note that the use of Gini tr. and IV tr. is described in Section 2.2

the sign A in the Solus register (meaning described in Section 3.1), is positive and in absolute value quite high in comparison with the other parameters. That means, that for clients with this property, the log of odds will be much higher, and hence the chance of being bad will be higher. This makes sense, because the Solus information is in general negative. We can see, that for married clients there is much lower chance to be bad since the parameter is negative and in absolute value also quite high. This coincides with the intuition based on Figure 3.2. The three variables left are all numeric and the parameters are all small negative. That means that with higher values of these variables the chance of being bad will decrease. This coincides with the intuition based on Figure 3.3.

Binning	Gini tr.	IV tr.	Stepwise	GINI coef.	accuracy
NO	NO	NO	NO	0.3883	0.6343
NO	NO	NO	YES	0.3883	0.6338
RP	NO	NO	NO	0.3765	0.6324
RP	NO	NO	YES	0.3790	0.6326
NO	NO	0.1	NO	0.3814	0.6283
NO	NO	0.1	YES	0.3812	0.6279

Table 3.3: Overview of characteristics obtained via cross-validation.

Variable	β
AGE_AT_TIME_OF_REQUEST	-0.0179
NRKI_RESULT	-0.0014
SolusA YES	0.8997
CLIENT_EMPLOYMENT_LENGTH	-0.0118
CLIENT_FAMILY_STATUS MARRIED	-0.2959

Table 3.4: Selected 5 most significant variables from the best logistic regression model.

At the end, we try to run the model on the whole dataset of approved loans. We obtain, that accuracy is 0.6380, sensitivity is 0.9358 and specificity is 0.2219, which is consistent with values computed on validation data.

3.2.3 Neural networks results

For neural networks purposes, we will not categorize the numeric variables. It is reccomended to scale them into $[0, 1]$ interval, but we do not need to do that, since it is done automatically by the function we use. To choose the architecture and parameters, we will try several options and observe the behavior. Hidden layer options from the set $\{13, 50, 100, 162, (50, 50), (100, 60), (100, 100), (150, 100), (200, 200), (40, 40, 40)\}$ were tried.

We decided to include only results with rectifier activation function (see Equation 1.11). Also hyperbolic tangent was tried, but the results were poor in comparison to rectifier. Both activation function options with and without dropout were tried. We used the ADADELTA method described in the theoretical part with values of $\rho \in \{0.9, 0.92, 0.94, 0.96, 0.99\}$ and $\epsilon \in \{10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}\}$. Then we chose 3 models with highest Gini coefficient reported on validation data and we did 10- fold cross validation on them in order to see whether they really behave good. Results from this exercise can be seen in Table 3.5 and Table 3.6.

Activation	Hidden	Epsilon	Rho	GINI coef. valid.
RWD	100	10^{-8}	0.99	0.3901
RWD	162	10^{-9}	0.99	0.3942
RWD	(150,100)	10^{-7}	0.99	0.3899

Table 3.5: Best 3 NN models w.r.t. GINI coef., no cross validation. RWD stands for rectifier with dropout.

In Table 3.7 we can see the scoring history for the best model. We can see, that the improvement of validation logloss is not very good. On the other hand, as we can see in Figure 2.2, it is not completely wrong, since the value of logloss around 0.6 corresponds to predicted probability of default around 0.55 in case that default occurs.

In Table 3.8 we can see 5 most important variables for the best model.

Again, the first thing we can see is, that all the variables we have in Table 3.8 we discussed in Section 3.1 as the distributions for good and bad clients is different so it is expected that they are important. As the structure of NN model is more complex than the logistic regression, the interpretation here is not straightforward. The result is slightly different from the logistic regression, we still have NRKI_RESULT, SolusA and AGE_AT_TIME_OF_REQUEST in the five most important variables list, but instead of the other two we have HOUSING and SolusD. An important thing to note here is, that the way we selected these variables to be the most important is different from the logistic regression case. In the logistic regression model, they are important in the sense of statistical significance. In the NN model the importance is derived according to method described in Section 1.2.4. The nice thing here is, that even though it is derived differently, it coincides quite a lot, which confirms it is reasonable.

For the best model, we have sensitivity of 0.7131 and specificity of 0.4070 on validation data, which is not perfect, but much better than in case of logistic regression. Sensitivity is lower, but this is not an issue, since it is still quite

Activation	Hidden	Epsilon	Rho	GINI coef.	accuracy
RWD	100	10^{-8}	0.99	0.3780	0.5727
RWD	162	10^{-9}	0.99	0.3808	0.5846
RWD	(150,100)	10^{-7}	0.99	0.3645	0.5742

Table 3.6: Best 3 NN models w.r.t. GINI coef., 10-fold cross validation results. RWD stands for rectifier with dropout.

epochs	training logloss	validation logloss
0.00		
1.07	0.62	0.63
4.31	0.61	0.62
11.86	0.61	0.62

Table 3.7: Scoring history for NN with hidden = 162, epsilon = 10^{-9}, rho = 0.99

	variable	relative importance	scaled importance
1	NRKI_RESULT	1.00	1.00
2	HOUSING OWN	0.96	0.96
3	SolusA NO	0.93	0.93
4	AGE_AT_TIME_OF_REQUEST	0.93	0.93
5	SolusD NO	0.83	0.83

Table 3.8: Variable importances (5 most important) in NN with hidden = 162, epsilon = 10^{-9}, rho = 0.99

high and sensitivity is not that important. More important is, that specificity is almost two times higher.

At the end, we run the model on the whole dataset and we obtain accuracy of 0.6639, sensitivity of 0.8196 and specificity of 0.4463, which is again consistent with the values obtained on validation data. That means, that the behavior of the model on train and validation dataset is very similar.

3.2.4 Decision trees results

Similarly as for neural networks, we will not bin numeric variables for random forest purposes. There are basically two parameters to tune while growing the forest - number of trees and sample rate per tree, which is maximal ratio of columns used for growing each tree in the forest. We tried number of trees from $\{50, 100, 200, 500\}$ and sample rate per tree in $\{0.2, 0.5, 0.8, 1\}$. Note that the algorithm used here is CART, but the feature and split selection rule is minimizing the residual sum of squares in the subtree, which means, that even if we want to have classification trees, the algorithm behaves like we had regression trees. It estimates probability of being in one of the classes in a single tree. Then the other probability from each tree is extracted as complement to one.

The best 3 models with respect to Gini coefficient can be seen in Table 3.9. For these models also 5-fold cross validation was performed, results are shown in Table 3.10.

Trees	Col. sample rate per tree	GINI coef. valid.
200	1	0.3833
500	0.8	0.3868
500	1	0.3857

Table 3.9: Best 3 RF models w.r.t. GINI coef., no cross validation.

Trees	Col. sample rate per tree	GINI coef.	accuracy
200	1	0.3727	0.5682
500	0.8	0.3723	0.5788
500	1	0.3757	0.5693

Table 3.10: Best 3 RF models w.r.t. GINI coef., 5-fold cross validation results.

In Table 3.11 we can see the scoring history for the best model. We can observe, that in contrast to NN, validation logloss improves faster than training logloss. They both end at the value of 0.62, so the result is similar to NN.

In Table 3.12 we can see the 5 most important variables in the best RF model.

We can see, that again we have AGE_AT_TIME_OF_REQUEST and NRKI_RESULT among the five most important variables. This is what we already expect. The interesting point here is, that the other three variables are not even plotted in Section 3.1, which means that we would expect that the discrimination power of those variables is not very good. This is very interesting, since the model gives good results. Again, we have to notice here, that the way of computing the importance is different to the previous two, it is done according to the method described in Section 1.2.3.

For the best model, we have sensitivity of 0.7135 and specificity of 0.4042 on validation data, which is very similar to the result we obtained using NN model.

	number of trees	training logloss	validation logloss
1	0.00		
2	1.00	12.58	12.55
3	2.00	10.40	5.32
4	3.00	8.96	2.72
5	4.00	7.74	1.74
6	5.00	6.51	1.21
7	6.00	5.42	0.94
8	7.00	4.71	0.81
9	8.00	3.99	0.78
10	9.00	3.37	0.75
11	10.00	2.79	0.72
12	11.00	2.40	0.70
13	12.00	2.05	0.69
14	13.00	1.77	0.68
15	14.00	1.59	0.67
16	15.00	1.39	0.67
17	16.00	1.26	0.66
18	17.00	1.14	0.66
19	18.00	1.05	0.66
20	19.00	0.98	0.65
21	20.00	0.93	0.65
22	21.00	0.89	0.65
23	22.00	0.85	0.64
24	23.00	0.81	0.64
25	74.00	0.64	0.63
26	133.00	0.63	0.62
27	249.00	0.62	0.62
28	460.00	0.62	0.62
29	500.00	0.62	0.62

Table 3.11: Scoring history for RF with 500 trees and col. sample per tree equal to 0.8

	variable	relative imp.	scaled imp.
1	CLIENT_REGION_PERMANENT_ADRESS	177882.42	1.00
2	AGE_AT_TIME_OF_REQUEST	141085.31	0.79
3	NRKI_RESULT	117248.38	0.66
4	AGREED_LOAN_AMOUNT	106239.28	0.60
5	AGREED_MONTHLY_REPAYMENT	102089.48	0.57

Table 3.12: Variable importance (5 most important variables) in RF model with 500 trees and col. sample per tree equal to 0.8

At the end we run the model on the whole dataset and we obtain accuracy of 0.8891, sensitivity of 0.9414 and specificity of 0.8159. All these values are very high. This means, that the model behaves quite good at validation data (in comparison with the logistic regression model much better and very similarly in comparison with neural network model) and even much better at train data. The overview of accuracy, sensitivity and specificity values for the three compared models is provided in Table 3.13.

	Logistic regression	Neural network	Random forest
accuracy - validation	0.6343	0.5846	0.5788
sensitivity - validation	0.9371	0.7131	0.7135
specificity - validation	0.2289	0.4070	0.4042
accuracy - whole dataset	0.6380	0.6639	0.8891
sensitivity - whole dataset	0.9358	0.8196	0.9414
specificity - whole dataset	0.2219	0.4463	0.8159

Table 3.13: Overview of accuracy, sensitivity and specificity values for the three compared models.

3.2.5 Overview of selected important variables

Let us now have a look at the most important variables with respect to all models and discuss a bit more their behavior. These are NRKI_RESULT and AGE_AT_TIME_OF_REQUEST, which are the most significant in logistic regression and also important in NN models, and CLIENT_REGION_PERMANENT_ADRESS, which is the most important in RF model.

In Figures 3.5 and 3.6 we can see distribution of defaults with respect to NRKI_RESULT and AGE_AT_TIME_OF_REQUEST, respectively. The red line is a logistic curve coming from logistic regression model based on only the one particular variable. Provided Gini coefficient is based on this model. In Figure 3.7 we can see distribution of CLIENT_REGION_PERMANENT_ADRESS for good and bad clients.

As we can see, the discrimination power of the most important variables themselves is not very good, and so the results we obtained are expected, because it is better when we use more variables, but not that much.

In the ideal case, in case of numeric variables we would like to see roughly speaking all or at least the most bad observations on the left and all or at least the most good observations on the right, or vice versa. So that the logistic curve can than be almost constant zero on one side and almost constant one on the other side. In the case of categorical variables we would like to see as much differences in the distribution of good clients and bad clients as possible.

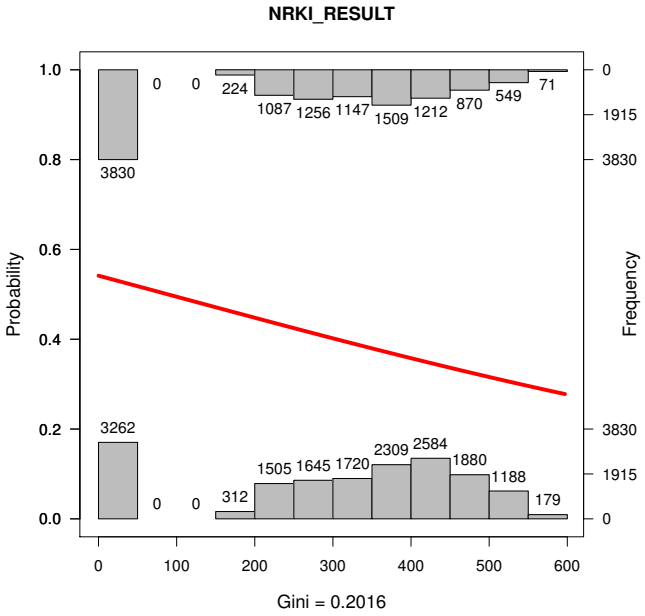

Figure 3.5: Distribution of NRKI_RESULT with logistic curve based only on this variable.

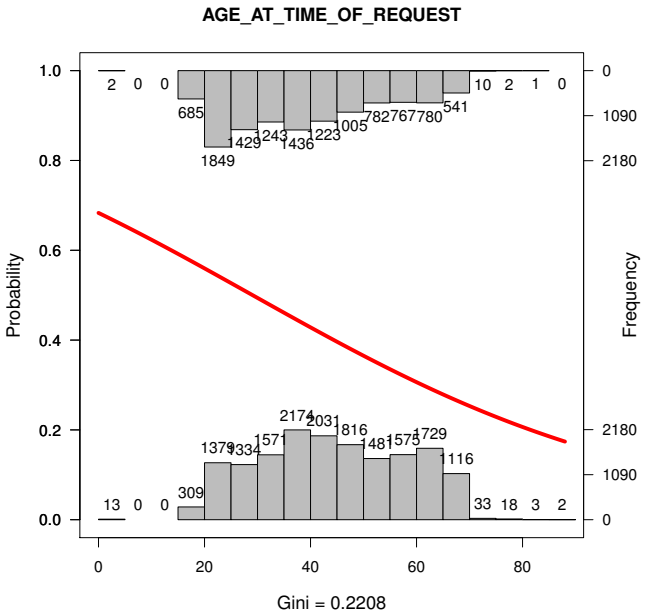

Figure 3.6: Distribution of AGE_AT_TIME_OF_REQUEST with logistic curve based only on this variable.

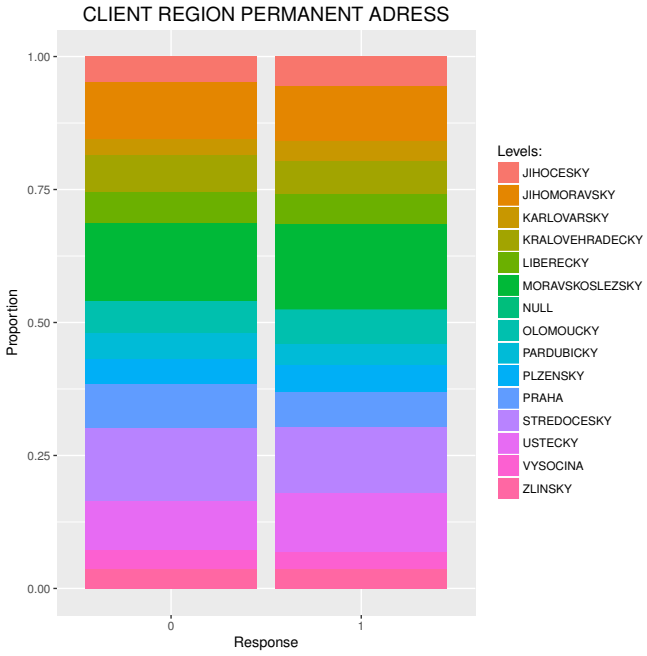

Figure 3.7: Distribution of CLIENT_REGION_PERMANENT_ADRESS for good and bad clients

3.2.6 Overall comparison

If we chose the best model only with respect to Gini coefficient, we would probably choose the logistic regression model. But all the models give quite similar values of Gini coefficient and it is not clear, which one is really the best one. At given dataset all the methods give us results of very similar quality measured by Gini coefficient.

According to accuracy, we would prefer the logistic regression model. But definitely not with respect to specificity, which is important and very low in case of the logistic regression model. Both neural networks and random forests behave very similarly in terms of accuracy, specificity and sensitivity computed on validation data. But if we compute these quantities on the whole dataset of approved loans, we obtain the best result using random forests. Note that these characteristics were summarized in Table 3.13.

As a part of dataset provided by PROFI CREDIT Czech, a.s., we also obtained a set of rejected loans. So now we will try 3 best models from the 3 methods on this data. We assume, that the criteria for rejecting the loans were reasonable, but we do not know them in particular. It is expected, that a reasonable model should predict default for most of these loans.

Logistic regression The logistic regression model provides us with predicted probability of default. We will consider the predicted value as prediction of default, if the probability is above 0.5. From the total number of 159 785, 60 592 was predicted as bad, which is 37.9%.

Neural networks From the neural network model, we obtain directly the predicted classification. From the total number of 159 785, 117 795 was predicted as bad, which is 73.7%.

Random forest From the random forest model, we obtain directly the predicted classification. From the total number of 159 785, 137 686 was predicted as bad, which is 86.2%.

So if we assume, that all the rejected clients are really bad, then we obtain the best prediction from the random forest model.

The bad rate in the dataset of approved loans we used for the modelling is 41.7%. So if we accept the logistic regression model which assigns approximately the same proportion of bad clients to the rejected dataset, it means, that the previous model which originally rejected the loans was making decisions almost randomly and the real proportion of bad clients in the population is around 40%. But according to very low specificity we do not believe this is true. The other two models assign to the rejected dataset approximately two times bigger proportion of bad clients than we observe in the approved dataset, which seems much more reasonable with respect to the original model, which we do not know, but we assume that it is reasonable, or at least it does not make decisions randomly.

Conclusion

The aim of this thesis was to summarize classical statistical methods used in credit scoring as well as the process of building a scoring model itself. This was done in the first part of the thesis.

Described methods were then applied on a real dataset provided by PROFI CREDIT Czech, a.s. Some practical issues were discussed in the second part of the thesis and then results of experiments were provided and discussed in the third part of the thesis.

The models were compared with respect to several measures of model quality. With respect to Gini coefficient, all three models give very similar results. The logistic regression model has quite good accuracy and very good sensitivity measured on both validation and whole data, but very low specificity, which is a problem, and hence we would not reccomend this approach. When we use the neural network approach, we observe similar value of accuracy, but lower sensitivity and higher specificity, again on both validation and whole dataset. According to the fact that specificity is more crucial than sensitivity we would prefer this model in comparison with logistic regression. Using the random forest, we obtain very similar accuracy, sensitivity and specificity on validation data as in case of neural network, but much better and very high values on the whole dataset, which means, that at train data the model uses the information maximally and behaves almost perfectly. Hence we would reccomend to use the random forest model.

Bibliography

I. A. Basheer and M. Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, (43):3–31, 2000.

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

N. J. Eck and L. Waltman. *VOSviewer: Visualizing scientific landscapes*. Leiden University, Netherlands, 2018. URL `http://www.vosviewer.com/`.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. 2001.

C. Fyfe. *Artificial Neural Networks and Information Theory*. 1.2. The University of Paisley, 2000.

T. D. Gedeon. Data mining of inputs: Analysing magnitude and functional measures. *International Journal of Neural Systems*, (2):209–18, 1997.

R. Hecht-Nielsen. Theory of the backpropagation neural network. *IJCNN*, (I): 593–605, 1989.

D. W. Hosmer and S. Lemeshow. *Applied Logistic Regression*. Second edition. John Wiley and Sons, Inc., Hoboken, New Jersey, 2000.

T. Hothorn, K. Hornik, and A. Zeileis. ctree: Cinditional inference trees. 2016.

B. Hssina, A. Merbouha, H. Ezzikouri, and M. Erritali. A comparative study of decision tree id3 and c4.5. *International Journal of Advanced Computer Science and Applications*, 4(3):13–19, 2014.

V. Kozmík. *Credit Risk in Banking*. 2006.

T. S. Körting. C4.5 algorithm and multivariate decision trees. 2018.

W-Y Loh. Classification and regression trees. 2014.

Microsoft. *SQL Server Management Studio*. Microsoft Corporation, Redmond, Washington, 2017. URL `https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-2017`.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

S. R. Safavian and D. Landgrebe. *A Survey of Decision Tree Classifier Methodology*. School of Electrical Engineering, Purdue University, 1990.

N. Siddigi. *Credit Risk Scorecards Developing and Implementing Intelligent Credit Scoring*. John Wiley and Sons, Inc., Hoboken, New Jersey, 2006.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, (15):1929–1958, 2014.

The H2O.ai team. *h2o: R Interface for H2O*, 2015. URL `https://github.com/h2oai/h2o-3`. R package version 3.1.0.99999.

M. D. Zeiler. Adadelta: An adaptive learning rate method. 2012.

List of Figures

List of Tables

List of Abbreviations

Gini tr. - Gini coefficient of one-variable model treshold
IV - information value
IV tr. - information value treshold
NN - neural network
RF - random forest
RP - recursive partitioning
RWD - rectifier with dropout
WOE - weight of evidence
WOE num. - WOE based binning performed on numeric variables only
SIS - Student Information System

A. Attachments

A.1 R code

The R code used for computations is attached in electronic form in SIS, file "A1_R_code.txt".

A.2 SQL code

The SQL code used for data preprocessing is attached in electronic form in SIS, file "A2_SQL_code.txt".

A.3 VOS viewer source files

The map file and network file used for creating the VOS viewer map are attached in electronic form in SIS. Can be used with VOSviewer [Eck and Waltman, 2018] directly to obtain an interactive map. Files "A3_VOS_mapfile.txt" (map file) and "A3_VOS_networkfile.txt" (network file).