Application Scorecard Modelling with Artifical Neutral Networks

Ana Miljkovic and Benjamin Chronéer

Master's thesis
2018:E19

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

Centrum Scientiarum Mathematicarum

Master's Theses in Mathematical Sciences 2018:E19
ISSN 1404-6342

LUTFMS-3342-2018

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/

Abstract

Credit scoring models, currently used for classifying new credit applicants, does often not have satisfactory predictive power and it is of high interest to find better models. With the recent surge of machine learning methods, artificial neural networks especially, many credit institutions are now curious of testing these methods in their fields. This thesis evaluates the practical use of artificial neural networks for credit score modelling. The practises in current credit scoring models used, the theory of artificial neural networks and a suitable development approach is discussed. It is found that artificial neural networks can outperform both current credit scoring models and other machine learning methods, such as the random forest. The main contribution of this thesis is that the networks are found to be reasonably transparent after applying a white-boxing method, but perhaps not transparent enough to comply with credit regulations. It is suggested that credit institutions can use artificial neural networks in some internal extent.

Contents

Chapter 1

Introduction

An artificial neural network is a machine learning method that has been, during the last decade, recognized as one of the top performing methods in many different application areas such as image recognition and natural language processing. Machine learning models can be used for prediction of future data by detecting and learning from patterns without a human telling it what to look for or in other ways supervising it. This thesis will investigate how artificial neural networks can assist in credit risk modelling.

Credit risk is the risk of a borrower not being able to pay back a credit and is by credit institutions measured with credit scoring models. This thesis will focus on credit scoring modelling for new applicants which is done with application scorecards and is especially difficult since the institutions have limited information about these applicants. One problem arising from applying artificial neural networks on application scorecards is that the model decisions can not be completely understood due to the model's black box nature that is, the model takes an input and gives an output decision although the basis for the decision is not explained. To combat this problem the model transparency might be increased through white-boxing methods.

This work will begin by describing the practises used for current credit score modelling followed by explaining the basics of artificial neural networks. With this as a background, the entire process from preparing raw data to extracting a probability of default from a constructed artificial neural network will be shown. The network performance will be showcased and discussed for three different data sets and the efforts of white-boxing the networks will be analyzed from a practical standpoint. Finally a conclusion of the viability of the use of artificial neural networks in the field of credit risk modelling is made and ideas for further work are discussed.

1.1 Objective

The objective of this thesis is to evaluate the practical use of artificial neural networks in the field of credit risk modelling, more precisely the modelling of an application scorecard. The modelling will be based on data sets consisting of credit card and loan applications.

Additionally the thesis will explore the interpretability of artificial neural networks and to what degree they can be explained. The thesis targets will be an audience with an engineering background or experience in the field.

1.2 Limitations

It was chosen to limit the thesis to application scorecard modelling as these models have less predictive parameters to utilize which in turn makes the default predictions problematic for current models. Stability analysis on the models' performance was not done since the idea was to have a model that can be updated daily or weekly and will not have to be able to make good predictions in the far future, which makes the issues arising from stability problems less of a factor. While artificial neural networks could be used to enhance data before predicting risk, for example by imputing missing values, this is deemed to be too extensive to include in this thesis and is instead left as further work.

Chapter 2

Credit Scoring

The main business of a credit institution is to make a profit by lending money without taking too great of a risk. The cost of providing a credit is know as expected loss (EL) and is mostly calculated as the expected money lost due to a borrower defaulting as

$$EL = PD \cdot EAD \cdot LGD. \tag{1}$$

PD is the Probability of Default, that is the likelihood that a credit will not be repaid. EAD is Exposure At Default which is an estimation of the total value that the lender is exposed to at the time of the credit default. LGD is the Doss Given Default, the fraction of EAD that will not be recovered if a credit default occurs. [Bolton et al., 2010] The risk for a borrower not being able to pay back credit obligations is by credit institutions measured with credit scoring models. Credit scoring models measures the probability of default which is used to assist the lender in determining if a credit should be approved or not. Credit scoring is done for new applicants using an application scorecard and for existing customers with a behavioural scorecard. A scorecard assigns each credit applicant a score that is translated into a probability of default.

Different commercial, regulatory and strategic areas have different requirements on credit scoring models. These requirements determine the development process and can for instance be to have the target value as the PD and a general acceptance of the model. Target value PD means that the model output should be expressed as a default probability and acceptance is the understanding of the rating results. The rating model should accurately estimate the creditworthiness and the model decisions should be easily understood. [Thonabauer and Nösslinger, 2004] The development of a credit scoring model can be divided into three steps. First, data gathering and preparation, then, the model development and lastly, the model validation.

2.1 Data Gathering and Preparation Practises

Before data gathering it is important to define what information should be gathered. There are two kinds of data to gather, quantitative and qualitative.

Quantitative data refers to for example credit card transactions or bank account activity, which is measurable numerical values. Qualitative data is obtained directly from the applicants by inquiring personal information. Data can be gathered either within the credit institution or externally. [Thonabauer and Nösslinger, 2004]

The data should be representative of the credit model's area of application and should include applicants with similar characteristics with both good and bad outcomes. From the definition of default used in *Basel* II, bad borrowers are defined as *the obligor being unlikely to repay its debt or by being 90 days past due or more on any payment to the credit institution.* [Basle Committee on Banking Supervision, 1998] The usual time-horizon for application scorecards is twelve months, past this time the customer is transferred to a behavioural scorecard.

The difference between behaviour scores and application scores is the borrower's own behaviour pattern as a parameter in the behavioural scorecards. If and when the payments are received every month demonstrates the borrower's willingness and ability to pay on the credit which is a very good predictor of future behaviour. Behavioural scorecards are thus more predictive than application scorecards which are only based on information available during the application such as age, salary, balance, security and credit history. [StatSoft, 2013]

Before developing the model a decision regarding parameter relevancy to the output must be made. Irrelevant inputs are excluded and some parameters might need to be divided into intervals to sustain model stability and acceptance. An example of a parameter that could be divided into intervals is age. This is done to avoid unintuitive model decisions such as believing that being exactly 35 years old infers a much higher credit risk than being 34 or 36 years old. Such a decision might have statistical basis and the performance might increase but the loss in acceptance is too great. If the input parameters are correlated it can lead to stability problems when estimating the scoring model weights. One rule of thumb is to only include parameters with a correlation less than 30%. [Thonabauer and Nösslinger, 2004] If there are missing values in the data, they can be either removed or replaced with for instance the mean value, a random value or with some other technique. It is also important to manage potential error values and outliers. Error values are values that are obviously incorrect, for an example a negative age or an age of 200. Outliers are values that do not follow the general trend of the data. [The Pennsylvania State University, 2018] Both of these can disturb the forecasting abilities of the model. Once the data is gathered and cleaned it is divided into a training sample, a validation sample and a testing sample. The training and the validation sample is used to find the optimal model weights and possible hyperparameters. The test sample is an out-of-time sample used for evaluating the model.

If the data set used in the model development only consists of accepted applicants a selection bias can occur. Selection bias is a variant of the missing value problem since the outcome value is missing for applicants that were rejected or never took up the credit. Developing a model based on only outcomes from the accepted applicants composes a skewed view of reality. An approach for avoiding selection bias is reject inference. The aim is to estimate the missing values, that is to classify whether the applicants would have been good or bad borrowers if they had been accepted or taken up the credit. Parcelling is a reject inference method based on simulating missing outcomes with a credit scoring model built on only accepted applicants. Then a new model is built using simulated and real outcomes but weighted to assign less importance to the inferred outcomes. [Missaoui, 2018] Parcelling is illustrated in figure 2.1.

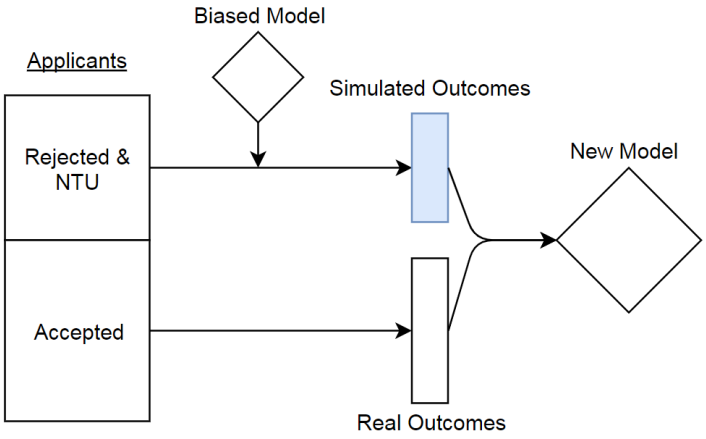

Figure 2.1: Illustration of the reject inference method parcelling. Missing outcomes for applicants that were rejected or never took up the credit (NTU) are simulated with a credit scoring model (biased model) built on only accepted applicants. A new model is built on both real and simulated outcomes.

2.2 Rating Model Development

Two types of credit scoring models are heuristic and statistical. The heuristic models attempt to use knowledge of the lending business to subjectively make decisions that reflects the model creators appraisals and one type of heuristic model is the classic rating questionnaires. In these, the scores given are set with intuition to score the significance of the inputs according to the creators. Instead of relying on the subjective experience of credit experts, statistical models are a great alternative as they set out to determine the optimal classification process using statistical procedures and data. Statistical models consist of a scoring function that assigns each applicant a score depending on the input parameters and the aim when developing the model is to find the optimal scoring function coefficients (weights), often estimated with maximum likelihood. Statistical models are very common in the lending business as long as there is enough data to find statistical significance and accurate data that fairly represents the field it is to be implemented in. Heuristic and statistical models are often used together in hybrid models. [Thonabauer and Nösslinger, 2004]

2.2.1 Regression Models

Current statistical models used in credit scoring are regression models that can either be linear or logistic where the logistic models often are preferred because of their probabilistic structure. As the outcome, Y_i, of an applicant is either default or no default, it can be seen as a binary stochastic Bernoulli distributed variable.

$$Y_i \sim Bernoulli(p_i) \tag{2}$$

Where p_i is the probability that Y_i takes on the value 1 and $1 - p_i$ is the probability that it takes on the value 0, that is

$$P(Y_i = y_i) = p_i^{y_i}(1 - p_i)^{1-y_i}, \tag{3}$$

where y_i is either 1 or 0. Logistic regression obtains a model score from the logistic score function

$$p_i(x) = \frac{1}{1 + e^{-\beta_0 - \sum_{j=1} \beta_j x_j}} \tag{4}$$

where the model score $p_i(x)$ is a value between zero and one, x is the input parameter vector and β are the scoring function coefficients. [Freedman, 2009] A model score can be interpreted as the credit risk of the applicant and can either be utilized in ranking the applicants between each other or for calculating individual default probabilities through risk class segmentation.

2.2.2 Risk Class Segmentation

The method of translating a credit score into a probability of default is called risk class segmentation and helps the credit institutions make decisions regarding the credit approval according to their risk profile. In mathematical notation this translation can be expressed as going from

$$P(Y_i = 1|Model, Data) \tag{5}$$

to

$$P(Y_i = 1|Data) \tag{6}$$

where *Data* is referring to the input data and Y_i is the binary realized outcome that is, default (1) or no default (0). The translation is often done by dividing the model scores in different risk classes and it is suggested to have 7-20 different ones. [Thonabauer and Nösslinger, 2004] By observing the default frequency in different model score ranges for the training sample, risk classes with an associated probability of default can be set up. Optimally each class should have an equal population and applicants with a probability of default over a certain cut-off point are all considered bad borrowers and can be assigned the same risk class. [Thonabauer and Nösslinger, 2004] The individuals with the lowest probabilities are assigned to the lowest risk classes. Depending on the distribution of the model score and what default probability fidelity is required, the risk classes can be adjusted.

2.3 Validation

Validation is the part where the developed model is evaluated by examining the model performance and stability as well as testing model predictions against realized outcomes.

2.3.1 Qualitative Validation

Validation can be separated into two areas, qualitative and quantitative validation. The goal of the qualitative validation is to ensure that the models are properly applied in practise which can be done with three overlying controls. [Thonabauer and Nösslinger, 2004]

- The model design control is validation of the model's documentation and transparency.
- The data quality check regards the completeness of data, amount of data, the representativity of the data and the raw data transformations.
- The use test regards internal use. This step controls that the integration of the rating procedures in the lenders risk management and reporting systems are implemented correctly.

2.3.2 Quantitative Validation

The quantitative validation includes the procedures in which the statistical metrics for the rating models are calculated. The metrics that describe the discriminatory power, the calibration accuracy and the stability, are reviewed by back-testing on previous data as well as bench-marking on separate data. Often a trigger level is set on which, if the performance metric falls below, should indicate a bad model fit. [Thonabauer and Nösslinger, 2004]

2.3.2.1 Discriminatory Power

The term refers to the model's classification accuracy. That is, its ability to separate the non-defaulting borrowers from the defaulting ones. In a binary classification problem the model outcomes will be labeled either positive (0) or negative (1). If the model correctly classifies the outcome it is labeled true, otherwise false. For unbalanced data, as credit default data tends to be, a metric like accuracy which is the fraction of correct classifications, is not usable. The model could simply guess that no applicant will default and still get a decent performance since the default rate is often very low. Accuracy is defined as

$$accuracy = \frac{\sum TP + \sum TN}{\sum T} \tag{7}$$

where TP is the true positives, TN true negatives and T total classifications. Suitable metrics are for instance the Gini Coefficient or the AUC-ROC. [Davis and Goadrich, 2006] The Receiver Operator Characteristic (ROC) curve considers the label imbalance and plots the True Positive Rate TPR versus the False Postive rate FPR and they are defined as

$$TPR = \frac{\sum TP}{\sum P} \tag{8}$$

where P is the total positives and

$$FPR = \frac{\sum FP}{\sum N} \tag{9}$$

where N is the total negatives. Thresholds are set as a monotonously increasing vector of values and for each threshold both the FPR and the TPR are calculated.

The metric extracted from the ROC-curve is called the Area Under Curve (AUC) and is calculated as the area under the curve using the trapezoidal rule. The Gini Coefficient is a similar metric and can simply be calculated as

$$Gini = 2AUC - 1 \tag{10}$$

An issue with these metrics is that their one dimensionality destroys some information about the shape of the ROC-curve which can convey some useful information especially in the case when two models get the same performance. [Davis and Goadrich, 2006] Figure 2.1 illustrates two ROC curves which appear to be close to optimal since the aim is to be in the upper-left-hand corner, in other words to have 100% true positive rate and 0% false positive rate. The ROC curve should be concave, that is curved to the right in the whole field, otherwise there might be a classification accuracy problem. The ROC-curve is often shown with a 45 degree line from (0,0) to (1,1) which is the result of a model randomly ranking the applicants. [Davis and Goadrich, 2006]

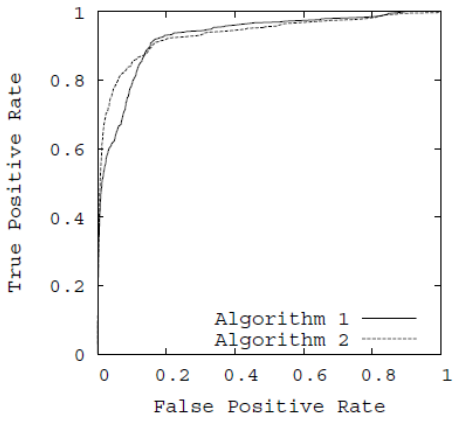

Figure 2.1: Two ROC curves which appear to be close to optimal since the aim is to be in the upper-left-hand corner, in other words to have 100% true positive rate and 0% false positive rate. From [Davis and Goadrich, 2006]

2.3.2.2 Calibration Accuracy

While the discriminatory power is a measure of the classification ability, the aim of the calibration is to assign the correct probability of default to each model score, y in equation 4. The calibration quality is measured upon how well the estimated PD agrees with the observed default rate. The default probabilities can be divided into different risk classes which are often concentrated on lower probabilities since higher resolution is desired in this area. Most commonly the data sets used in calibration are the default probabilities predicted over a twelve month time-horizon, their actual outcome and the amount of applicants in each risk class. [Thonabauer and Nösslinger, 2004]

The Hosmer-Lemeshow goodness-of-fit test is suitable to evaluate if the risk classes group the default probabilities appropriately. The test statistic is calculated as

$$HL = \sum_{i=1}^{N} \frac{(O_i - E_i)^2}{n_i \pi_i (1 - \pi_i)}, \tag{11}$$

where O is the observed defaults, E is the expected defaults, n is the population in each risk class, π is the predicted risk and N is the amount of risk classes. The test is evaluated as a Chi-Squared distributed value with $N - 2$ degrees of freedom. [Chen et al., 2014]

2.3.2.3 Stability

Stability is consistency over time and models' performance should be robust against a changing landscape. There are two aspects to consider and the first one is various forecast horizons. When models are optimized for say standard twelve months their discriminatory power will obviously get worse with longer time horizons. This decrease in discriminatory power should be reasonably stable and limited, and if not, it could be an indicator of a model prone to failure in the future. The second aspect is the general conditions for model use. External conditions such as the economic and political environment can substantially affect the default outcomes. There are also internal conditions for the lender such as changes in business strategies that can affect performance of the rating models. A metric to measure whether or not a model is stable is the population stability index (PSI). This metric ensures that the percentage of population assigned a certain risk class stays approximately the same over time. The PSI is calculated as

$$PSI = (C - E) \cdot \ln(\frac{C}{E}), \tag{12}$$

where C stands for the current population fraction in a score class and E for the expected population fraction in that same class. [Thonabauer and Nösslinger, 2004]

Chapter 3

Artificial Neural Networks

3.1 Introduction

It is of high interest to apply artificial neural network algorithms in application scorecard modelling since they are expected to perform better than current models used due to the increased model complexity. Artificial neural networks can find identifying linear and nonlinear patterns and features in the data. The universal approximation theorem states that networks of sufficient complexity can approximate any continuous function. [Goodfellow et al., 2016]

3.2 Elements of an Artificial Neural Network

A neural network is an architecture that consist of an input layer, any number of hidden layers and an output layer. The layers consist of nodes (neurons) and weights between them. The most common network is the feed-forward neural network where the input signals enter from one end and propagates through the network to the other end, shown in figure 3.1.

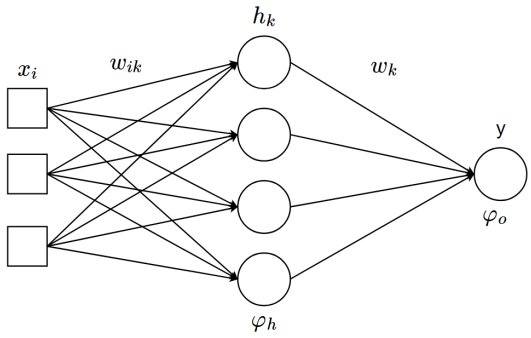

Figure 3.1: A standard feed-forward network with input layer, x, one hidden layer, h, and one output node, y. w_{ik} is the weight between the input node i and the hidden layer node k. w_k is the weight from the hidden node k and the output node. φ_h is the activation function and φ_o is the output function.

The output signal in figure 3.1 is calculated as

$$y = \varphi_o\Big(\sum_k w_k h_k\Big) = \varphi_o\Big(\sum_k w_k \varphi_h\Big(\sum_i w_{ik} x_i\Big)\Big) \tag{1}$$

where w_{ik} is the weight between the input node x_i and the hidden node h_k. In every layer there is an added bias node which is not shown in the figure. $\varphi(x)$ is the propagation function and this will be different depending on purpose and position in the network. [Goodfellow et al., 2016] Some examples are

1. Linear function
$$\varphi(x) = x \tag{2}$$
2. Threshold function
$$\varphi(x) = \left\{ \begin{array}{ll} 0, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0 \end{array} \right\} \tag{3}$$
3. Logistic function
$$\varphi(x) = \frac{1}{1 + e^{-x}} \tag{4}$$
4. Hyperbolic tangent function
$$\varphi(x) = \tanh(x) \tag{5}$$
5. Rectified function (ReLu)
$$\varphi(x) = \left\{ \begin{array}{ll} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{array} \right\} \tag{6}$$
6. Softplus function
$$\varphi(x) = \log(1 + e^x) \tag{7}$$

7. Softmax function

$$\varphi(x) = \frac{e^{a_i}}{\sum_j e^{a_j}} \tag{8}$$

$$a_i = \sum_k w_{ik} x_k \tag{9}$$

The logistic and the hyperbolic tangent function are the most commonly used functions in the hidden layers. The output function is dependent on the problem and for regression problems, the linear function can be used. For classification problems the logistic or softmax function is more suitable as classes can be defined as zeros and ones. The softmax function is used when there are multiple output classes. [Goodfellow et al., 2016]

3.3 Learning

The objective of the network is to correctly classify data, in other words map the output y, which is a function of the input parameter vector $\mathbf{x}$ and the weight vector $\mathbf{w}$, to the target label d. The target label d is the known correct class of a certain input pattern, $\mathbf{x}$, and each input pattern consist of parameters where each parameter corresponds to an input node. Training data sets consists of many input patterns and learning in this context, refers to the updating of the weights, $\mathbf{w}$, that lead to the solution $y = d$ for all input patterns. In a classification problem the difference between y and d is minimized by minimizing the cross-entropy error function with regards to the network weights. For a two class problem the probability for observing either class is the Bernoulli distribution discussed in chapter 2.2.1

$$P(Y_i = d_i|\mathbf{x}) = y_i^{d_i}(1 - y_i)^{1-d_i}, \tag{10}$$

where $\mathbf{x}$ is the input parameter vector, y_i is the network output and d_i is the target label, 0 or 1, for input example i. The probability $P(Y_i = d_i|\mathbf{x})$ indicates how well the model has predicted the true label d_i and the aim is to maximize this probability which is done by using the principle Maximum Likelihood. The Likelihood of a data set is defined as

$$P(\mathbf{x}, d_i) = \prod_{n=1}^{N} P(x_n, d_i) = \prod_{n=1}^{N} P(Y_i = d_i|x_n)P(x_n) \tag{11}$$

and is a function of the weights, $\mathbf{w}$, since y_i is a function of the weights. Instead of maximizing the Likelihood one can minimize the function's negative logarithm and define the error function as

$$E(\mathbf{w}) = -\log\big(\prod_{n=1}^{N} P(Y_i = d_i|x_n)P(x_n)\big) = -\sum_{n=1}^{N} \log(P(Y_i = d_i|x_n)) - \sum_{n=1}^{N} \log(P(x_n)) \tag{12}$$

Since the last term is independent of $\mathbf{w}$ the error function can be rewritten as

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \log(P(Y_i = d_i|x_n)) \tag{13}$$

Inserting equation 10 gives the cross-entropy error function

$$E(\mathbf{w}) = -\sum_{i=1}^{N} \Big(d_i \ \log(y_i) + (1 - d_i) \log(1 - y_i) \Big), \tag{14}$$

which sums over the whole training set. The aim of the minimization is to find the weight vector that optimally predicts the classes. In practise, the network will divide the input space with hyperplanes into regions which group the different classes. It is the hidden nodes that work as feature detectors and divide the input space as shown in figure 3.1. [Goodfellow et al., 2016]

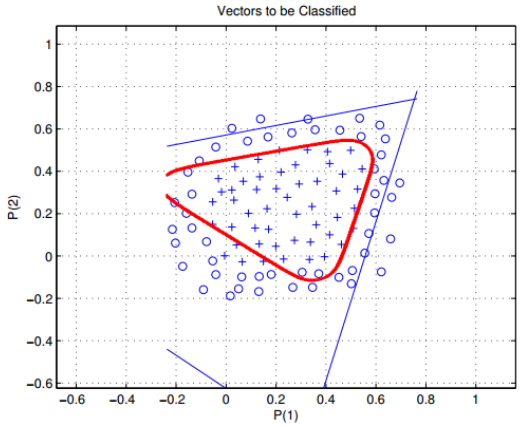

Figure 3.1: Illustration of an input space of two variables. With three hidden nodes the space will be divided by three lines, or hyperplanes in more dimensions. The red line is the boundary between the resulting classification regions. From [Ohlsson, 2017].

When the training data has several associated labels $\boldsymbol{d}$, the learning is called supervised. It is possible to train a network without labels, called unsupervised learning, where the network on its own decides identifying patterns and features in the input data. [Goodfellow et al., 2016] The most common method for supervised network learning is backpropagation which uses gradient descent algorithms to obtain the gradient

$$\frac{\partial E}{\partial w} \tag{15}$$

and find the optimal weights. As the name suggests the error is propagated backwards through the network and the weight updating is done one layer at a time. That is, the method works by first calculating the updated output weights and then calculating the updates of the weights backwards one layer at the time. The derivative for the output layer, L, of weights can be separated into the partial derivatives of the error function E, output function y, and the summed input a from equation 9 as

$$\frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial a^L} \frac{\partial a}{\partial w_j^L}. \tag{16}$$

w_j^L is the specific weight from node j to the output layer L.

When using a cross-entropy error function the derivative of E with respect to the output y is computed as

$$\frac{\partial E}{\partial y_i} = -\left(\frac{y_i - d}{y(1 - y_i)}\right) \tag{17}$$

with a minimum for $y = d$. For classification problems the logistic output function can be used which has the derivative

$$\frac{\partial y_i}{\partial a^L} = \frac{1}{1 + e^{-a^L}}. \tag{18}$$

The derivative of the summed input a with respect to the weight w_j is computed as

$$\frac{\partial a^L}{\partial w_j^L} = x_j^L, \tag{19}$$

where x^L is the node values in layer L. In the second to last layer the derivative is simply extended as

$$\frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial a^L}\frac{\partial a^L}{\partial x^L}\frac{\partial x^L}{\partial a^{L-1}}\frac{\partial a^{L-1}}{\partial w_{jk}^{L-1}}. \tag{20}$$

Where w_{jk}^{L-1} is the weights from node j to node k in layer $L - 1$. The calculations can proceed for as many layers the network has. [Goodfellow et al., 2016]

3.4 Gradient Descent

Gradient decent attempts to find the weights that minimizes the error function and the idea is to iteratively calculate the gradient and take a small step in negative direction until convergence. Depending on how much data is used to compute the gradient, the accuracy and the time it takes for a weight update can vary. If there are many layers, the resulting weight update is a product of many derivatives. There are three variants of gradient descent which differ in how much data is used. [Ruder, 2016]

3.4.1 Batch Gradient Descent

Batch gradient descent (BGD) computes the gradient for the whole training data set consisting of many input patterns as

$$\Delta w_{jk} = -\eta\left(\frac{1}{N}\sum_{n=1}^{N}\frac{\partial E(n)}{\partial w_{jk}}\right) \tag{21}$$

where η is the learning rate which determines the size of the step that is used to reach a minimum, n is an input pattern and N is the total amount of input patterns. This variant can be very slow since the gradients for the whole set are calculated to be able to perform just one update. BGD will converge to a local or global minimum for non-convex surfaces. [Ruder, 2016]

3.4.2 Stochastic Gradient Descent

Stochastic gradient descent (SGD) performs an update for each input pattern, n, and updates the weights as

$$\Delta w_{jk} = -\eta \frac{\partial E(n)}{\partial w_{jk}}. \tag{22}$$

Since the input patterns can vary quite a bit SGD updates the weights with a large variance which makes the error function fluctuate a lot. Initially, SGD is much faster than BGD and can in contrast, due to its fluctuation, jump to new and potentially better minima. A consequence that follows is that SGD can miss the minimum by overshooting instead of converging. A solution is to implement a slowly decreasing learning rate, η_i, which fulfills

$$\sum_i \eta_i = \infty \text{ and } \sum_i \eta_i^2 < \infty \tag{23}$$

and will result in the SGD almost surely converging to a local or global minimum for non-convex surfaces. [Ruder, 2016]

3.4.3 Mini-Batch Gradient Descent

Mini-batch gradient descent is a combination in between BGD and SGD where it performs an update for a mini-batch from N input patterns as

$$\Delta w_{jk} = -\eta \left(\frac{1}{P} \sum_{n=j+1}^{j+P} \frac{\partial E(n)}{\partial w_{jk}} \right), \tag{24}$$

where the mini-batch P is a subset of inputs often in the range $50 - 256$. [Ruder, 2016] By using a subset the variance of the weight update is reduced which can make the convergence more stable but still not guaranteed. Another challenge is to choose a suitable learning rate, if it is too small the convergence will be extremely slow but if it is too large it might not happen at all. Again, a necessary solution is a slowly decreasing learning rate. [Ruder, 2016]

3.4.4 Adam

There are many ways to enhance the gradient decent method and one such example is by using a dynamic learning rate. The idea is to change the learning rate to speed up or slow down the training process depending on the sign of the error changes. A popular extension of the dynamic learning rate is the ADAptive Moment estimation, Adam. Adam is considered one of the best gradient descent optimization algorithms and computes adaptive learning rates for every weight. [Kingma and Ba, 2014] Adam also keeps a running exponentially decaying average both of past squared gradients and of past gradients.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{25}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{26}$$

m_t is the estimate of the mean, v_t estimate of the variance of the gradients and g_t^2 indicates the element-wise square. The Adam update rule is

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t \tag{27}$$

where

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{28}$$

and

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{29}$$

It is proposed to have $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Other ways of improving the gradient descent is for instance to normalize the initial weights with zero mean and unit variance. With this approach larger learning rates can be used. [Ruder, 2016]

3.5 Overfitting

The performance of a network on training data can easily reach almost perfect discrimination power by just adding more parameters, weights and nodes, to the network. At some point, the network will start learning from noise in the data instead of focusing on the overall structure. This means that the network will learn every detail in the training data which will not generalize to new data and thus worsen the generalization performance. This is called overfitting and is illustrated in figure 3.1. [Ohlsson, 2017]

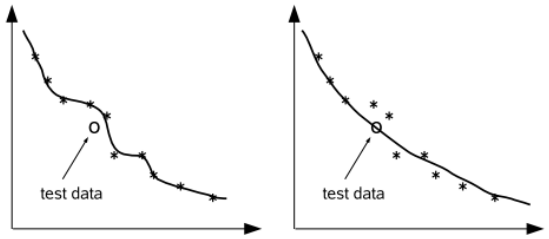

Figure 3.1: Left figure illustrates overfitting, very good fit to the training data, illustrated with asterisks, but not so good generalization performance, that is not so good fit to the test data. Right figure illustrates no overfitting, good generalization performance but a little worse fit to the training data. From [Ohlsson, 2017].

3.5.1 Regularization Term

Regularization is an approach to prevent overfitting. One cause of overfitting is having too many parameters, weights, with regards to the number of inputs to the network and a way to control this is by adding a regularization term to the error function as

$$\widetilde{E}(\omega) = E(\omega) + \alpha\Omega \tag{30}$$

where Ω is the regularization term and the amount of regularization is determined by the hyperparameter α. Ω can be set to

$$\Omega = \frac{1}{2}\sum_i w_i^2 \tag{31}$$

where the sum goes over all weights except for the bias weights. With this regularization term unnecessary weights will be forced to zero and the relevant (large) weights will become even more significant. This is called weight decay in machine learning or ridge regression in statistics. [Ohlsson, 2017]

3.5.2 Max L2 norm regularization

The maxnorm regularization introduces an upper bound for the combined size of the weights, $\hat{w}$, feeding into a specific node.

$$||\hat{w}||_2 = \left\{ \begin{array}{ll} ||\hat{w}||_2, & \text{if } ||\hat{w}||_2 < c \\ c, & \text{if } ||\hat{w}||_2 \geq c \end{array} \right\} \tag{32}$$

The constraining hyperparameter c is usually in the range of $2 - 4$ and if the norm of the weight vector is larger than this upper bound it is simply scaled down with $||\hat{w}||_2/c$. This regularization term restricts how large the weights can be and can prevent the network from exploding. [Srivastava et al., 2014]

3.5.3 Early stopping

The aim is to have as good of a generalized performance as possible, that is a low validation error. When overfitting occurs the validation error starts to increase even if the training error decreases. Therefore one can implement another regularization technique that is called early stopping where the idea is to stop the training of the network when the validation error starts to increase. This procedure is illustrated in figure 3.2. [Ohlsson, 2017]

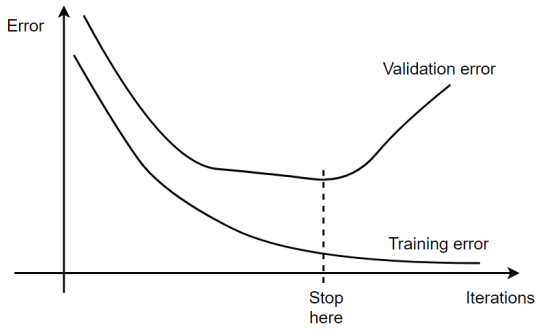

Figure 3.2: The early stopping method stops the training when the validation error starts to increase.

3.5.4 Dropout

The dropout method is able to efficiently avoid overfitting by temporarily disabling nodes and all their connected weights. Nodes can be dropped in all layers of the network except for the output layer. They are dropped with a probability of $1 - p$ where p is the probability to keep the nodes. This hyperparameter can vary for different layers and typically the nodes in the input layer has a higher chance of being kept. It is also possible to let the product np stay constant, that is the total number of disabled nodes should remain the same in each iteration instead of every node having the same independent probability. The former method however is more common and $p = 0.5$ for the hidden layers has been shown to yield close to optimal results. [Srivastava et al., 2014] Figure 3.3 illustrates an iteration of the dropout method. Nodes have randomly been selected to be inactive and the learning is done with the thinned network.

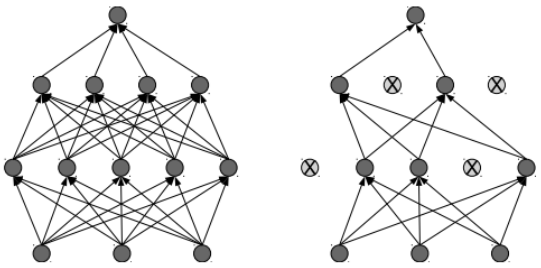

Figure 3.3: Illustration of the dropout method on a multilayer network. Some nodes are deactivated, for the next input pattern the deactivated nodes will change. With this method overfitting is avoided. From [Ohlsson, 2017]

The backpropagation in the learning process will work as usual. It is simply applied to the thinned network while the deactivated nodes are assumed a gradient of zero in each iteration.

Practically, the weights will be trained on an ensemble of different networks and will work as a model averaging which will reduce the model's performance variance. Dropout has been shown to significantly improve the generalization performance. [Srivastava et al., 2014] A practical detail to keep in mind is the fact that the expected input to any given node is larger in the model testing, where all nodes are active, compared to the training where some nodes are not active. To compensate for this a multiplication of p is applied to the outputs of all nodes.

To optimize the network further, additional regularization can be applied. Srivastava et al. suggest that norm constraining can work especially well with dropout. With this supposedly effective method of regularization one might think that it is better to design a network with many weights as the superfluous ones will be controlled by the regularization and not harm the performance. But it might be wise to be conservative because the time it takes to train a network with dropout is usually $2-3$ times longer than without dropout. [Srivastava et al., 2014]

3.6 Artificial Neural Network Architectures

The simplest architecture of a neural network is the perceptron which only has an input and an output layer. If hidden layers are added the structure is called a multilayer perceptron (MLP). There are many different kinds of network architectures and they mainly differ in how the network nodes are connected. Architectures such as convolutional and recurrent networks are can be used in for an example image recognition and natural language processing where the data is structured in a particular way. A combination of many single networks is called an ensemble of networks. The single networks, $y_i(x)$, can be combined by averaging their outputs as

$$y_{ensemble}(x) = \frac{1}{N}\sum_i y_i(x) \tag{33}$$

where N is the amount of single networks. The average error of an ensemble is the average error made by the individual networks minus the average variance of the ensemble members. This means that a good ensemble requires members that differ from each other. Different members can be produced by training networks with different initial weights but this may not create enough diversity. Instead K-fold splitting can be used which means that the training data set is equally divided into K parts. The ensemble member a is trained on the training data set that contains all K parts except for part a, the result will be K different networks trained on slightly different training data sets. This can be repeated several times by randomly splitting the data set into different parts. The dropout regularization method can be seen as an ensemble of different network architectures. [Goodfellow et al., 2016]

3.7 Network Interpretation by White-Boxing

Despite the often very good performance of neural networks in various application areas they are not always the obvious method of modelling. Because of their black box nature the decisions they make are rarely understood. Issues arising from this is untrustworthiness from users which in turn can limit practical implementation. To avoid these problems one would like to increase the models' transparency.

Understanding the model's decisions can also help detecting misclassifications or errors made while coding, for example by analyzing the principal components or finding undesirable correlations. This can for instance be that the network believes the ID number of an applicant is heavily correlated with default or that the color green in a image implies that there is a dog in it.

3.7.1 LIME

Riberio et al. proposes the LIME technique, short for Local Interpretable Model-agnostic Explanations, as a way to explain any decision made by a black box model. Local refers to explaining the model's behaviour around the decision and interpretable specifies that this explanation must be understandable for humans. Model-agnostic represents the fact that LIME can provide explanations for any black box model.

LIME desires to approximate a decision based on a specific combination of input parameters, x, made by the uninterpretable model (the ANN), f. This approximation is defined as a model $g \in G$, where G is a class of interpretable models which can serve as an explanation to the model output. The complexity of g is measured with $\Omega(g)$ and the local inaccuracy between $f(x)$ and $g(x)$ is represented with the function $\mathcal{L}(\cdot)$. The locality around x is defined as the distance measure π_x and the explanation obtained from LIME can be expressed as

$$\xi(x) = \underset{g \in G}{argmin} \quad \mathcal{L}(f, g, \pi_x) + \Omega(g). \tag{34}$$

$\xi(x)$ is the explanation model g that minimizes the local inaccuracy, $\mathcal{L}(f, g, \pi_x)$, without assuming anything about the model f since the explainer is model-agnostic. It is suggested to let G be a class of sparse linear models in which case the minimization search can be performed using perturbations of x. LIME learns how the model behaves by sampling from the neighbourhood of x to obtain decisions based on slightly different input parameter combinations, x'. The decision of the uniterpretable model, $f(x)$, is used as a target label for the decisions made by the interpretable model, $g(x')$ and by weighting the difference between the original decision, $f(x)$, and the simpler model's decisions in the neighborhood, $g(x')$, with the distance measure π_x the local inaccuracy is defined as the weighted square loss function

$$\mathcal{L}(f, g, \pi_x) = \sum_{x, x'} \pi_x(x')(f(x) - g(x'))^2. \tag{35}$$

The distance measure $\pi_x(x')$ is defined as the exponential kernel

$$\pi_x(x') = \exp\big(-D(x, x')^2/\sigma^2\big), \tag{36}$$

where D is the distance function, with width σ, which is picked depending on the classification problem. [Ribeiro et al., 2016]

For an example, if LIME should explain why a model has decided that a customer with an input parameter combination x has a high probability of default, LIME would start by changing input parameters in x to find each input's contribution to the decision. LIME would change one or several input parameters at a time by varying degrees, or remove them completely, to obtain different decisions and conclude each input parameter's importance.

In this example the explanation from LIME would be illustrated as the coefficients of the linear model, imagined as scores given to the input parameters depending on their contribution to the model score. The process of LIME is illustrated with a simplified example in figure 3.1.

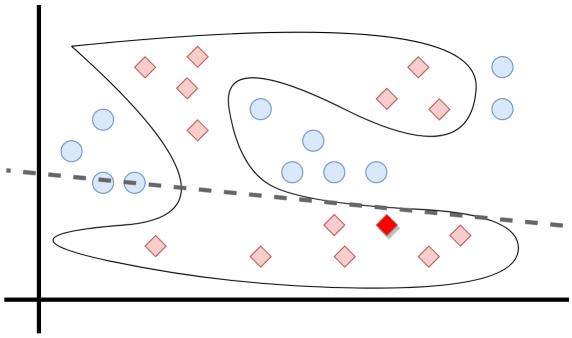

Figure 3.1: An illustration of the Local Interpretable Model-agnostic Explanations (LIME) technique. If the user wants an explanation for the classification of the marked diamond a local linear approximation of the decision boundary is made (the dashed line).

The blue and pink shapes represent two classes in an input space and the non-linear curve is the model's decision boundary f. The marked red diamond is the input parameter combination x and the sampling is made around it. The approach finds a linear function, g, that locally represents the model's decision boundary shown as the dashed line in figure 3.1. For large data sets it is wise to pick subsets of input parameter combinations that differ from each other to maximize information extracted and to avoid redundancy. LIME has been shown to be able to deliver a high trustworthiness compared to other white-boxing methods. [Ribeiro et al., 2016]

Chapter 4

Developing Artificial Neural Networks for Application Scorecard Modelling

Since artificial neural networks are problem dependent it is important to consider the network application when building the model. In this chapter an approach to the development of artificial neural networks is described and implemented. The implementation is done in Python with Keras and TensorFlow backend.

4.1 Defining the Problem

The first part of the process in designing the network is determining the objective, what should the output be and what error metric should be used. When considering application scorecards the final output is the probability of default which is a continuous variable between zero and one. An appropriate output function is therefore the logistic function which is bounded between zero and one. This output will be suitable for later risk class segmentation. The labels of the training data is binary, default or no default, and the frequency of defaults was approximately 1-2%. The problem formulation is to translate the classification of default and no default into a PD. Because the data is imbalanced, the error metric should take this into account. The loss function in the training algorithm is set to be a weighted binary cross entropy error function. This function weights the less frequent defaults higher to compensate for the skewness in the data. The error metric used in the validation is set to the AUC-ROC which measures the performance of the networks fairly by taking into account the imbalanced labels.

4.2 Analyzing and Preparing the Data

The data used was provided by a Nordic bank and the data analyzing and preparation process is illustrated in figure 4.1.

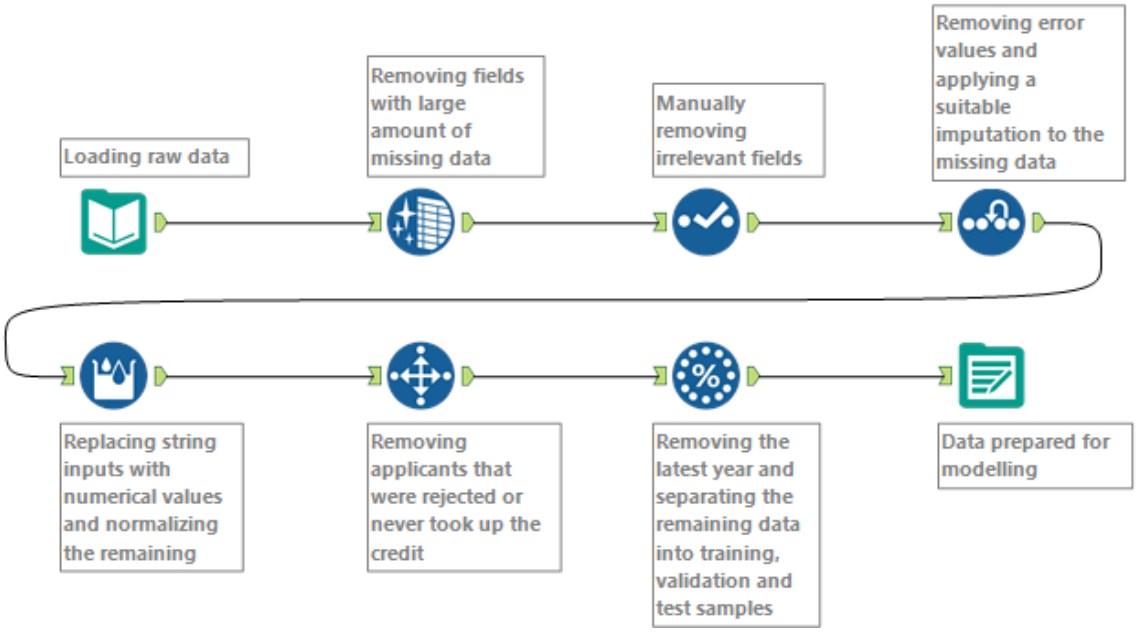

Figure 4.1: The data analysis and preparation process. First the data is loaded and inputs with a high amount of missing data is removed. Then, irrelevant inputs, such as application number, were removed. The remaining inputs were cleaned from error values and had their missing values imputed. Class data, such as marital status, was translated to numerical values. Finally the applicants which had no information whether or not a default had occurred, were removed and the data was separated into a training, a validation and a test sample.

The modelling was based on data sets consisting of different credit card applications as well as loan applications. These had a varying amount of information that could be used as inputs. To begin with, the inputs were analyzed based on their degree of completeness and their relevancy to the outcome, that is default or no default. Irrelevant inputs were removed, such as the ID-number and the application number. No consideration to input correlation was taken since the aim was to extract the maximum amount of information. Inputs consisting of strings, for an example gender, had these replaced with suitable numerical representations, in this case with -1 and 1 for women and men respectively. In the case of multi-class inputs, for example marital status, the categories were separated into a vector of inputs with binary values. Inputs that had a lot of missing values, around 50%, were removed due to the difficulties with replacing the missing data with suitable values. Inputs with fewer missing values had these replaced with the median. Obvious wrong values were removed from the inputs. Figure 4.2 shows the range of age which indicates the presence of at least one wrong value at age 745, which is obviously a mistake.

Figure 4.2: Age range in the data with min value, max value and the middle 50 percentile. There is at least one obvious error value at age 745.

Figure 4.3 shows the distribution of the input parameter age after removal of error values.

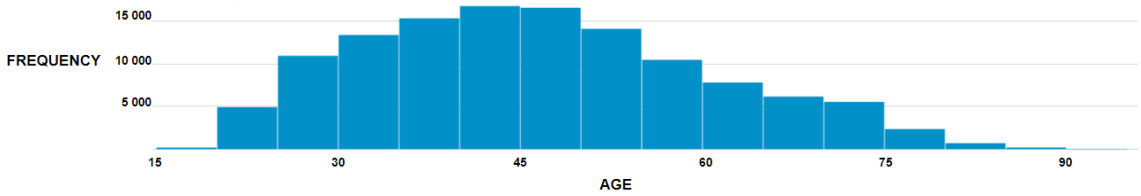

Figure 4.3: The age distribution in one of the data sets.

The remaining inputs were normalized with zero mean and unit variance to speed up the convergence of the gradient decent in the network training. Applicants that were rejected or never took up the credit appear as missing values in the labels, shown as [Null] in figure 4.4, and were removed from the data. It would be possible to use this data for reject inference as discussed in chapter 2 and was done at an early stage of the development, but it did not have an effect on the performance in this particular case and is left out from the result in chapter 5. Figure 4.4 also shows the overall default distribution.

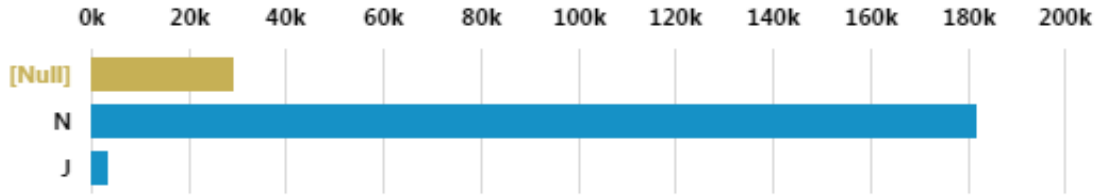

Figure 4.4: The default distribution for one of the data sets. N stands for no default and J for defaults. [Null] is the missing values which represents the applicants that never took up the credit or were rejected.

The latest year in the data was removed since the defaults are calculated on a twelve month time-horizon and not all defaults have been realized. The frequency of defaults are not constant over time and this raises questions on how to divide the data into a training, validation and test sample. The out of time test sample was chosen to be the latest 15-20% of the data. The remaining sample was shuffled to remove seasonality and then divided into training and validation, 80% and 20% respectively, figure 4.5 shows the arrangement of three different data samples.

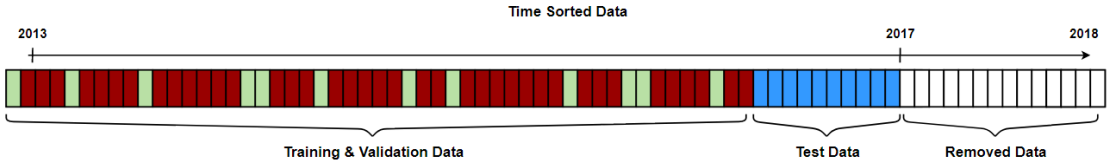

Figure 4.5: Overview of the data segmentation, sorted by time. The earlier data is used for training and validation in the model development. This data is shuffled to remove seasonal effects. The test sample is the later out-of-time set. The latest year is removed because not all defaults have been realized for this data.

4.3 The Architecture

The next step in the process was to set up a working network and decide upon a suitable benchmark to test against. The network was tested against three different models.

- **A Nordic bank's established application scorecard:** The scores given to the applicants by the Nordic bank was available and were used to get a ROC-curve for comparison. This model was developed in a time period before the data sets used.
- **Logistic regression:** A logistic regression model implemented with scikit-learn's python package. It was trained on the same data as the artificial neural network.
- **Random forest:** A more complex machine learning model known as random forest also implemented with scikit-learn's python package. Random forest has been shown to provide very good predictions for credit applications. [Blomgren and Vitestam, 2017] The only hyperparameter that was tuned was the amount of trees and was set to be 500 with no max depth. The model was trained on the same data as the network and the logistic regression.

Regarding the problem at hand, a fully connected feed-forward network was chosen as a suitable architecture. As for optimization methods the mini-batch gradient descent with Adam was used as it is computationally fast and precise and the parameters were set to the proposed ones in chapter 3. The activation and output functions were set to be the logistic function to introduce non-linearity to the model, as oppose to the linear or ReLu functions. The initial weights for all connections were uniformly distributed for all layers. As regularization, the dropout method was applied because of its simplicity and its remarkable efficiency. The dropout rate was set to 0.1 on the input layer and 0.5 on the hidden layers since this is close to optimal as discussed in chapter 3. This allows usage of a very large network without the consequence of overfitting. The number of hidden layers and the amount of nodes is somewhat arbitrary and was chosen by setting the initial number to be small and later be increased until no significant performance improvement was detected.

4.4 Iterating Changes

After the initial iteration the training error was evaluated. The training error should be sufficiently small with a decent model complexity, and if not, the number of nodes and hidden layers should be increased. With an acceptable training error the focus should shift towards improving the generalization performance. The common approach is to improve the network by tuning the hyperparameters in the learning algorithm. It is important to understand the impact of the hyperparameters to make intelligent adjustments manually. It is also possible to make an automatic hyperparameter search, but even in this case good initial values and ranges are important. In this development the hyperparameters in focus were the amount of nodes and the initial learning rate in the Adam gradient decent method discussed in chapter 3. The parameter search for the learning rate was done on a logarithmic scale to have a step-wise proportionally constant search. The number of hidden layers were chosen manually in the range $1 - 5$ with different proportions of a total of N nodes which in turn was found by a parameter search. The learning rate and the amount of nodes N were chosen based on the performance (AUC-ROC) on the validation sample.

4.5 Risk-Class Segmentation

Once a model score between zero and one was acquired it was translated into a probability of default through risk class segmentation. The score was divided into 14 different risk classes with equal population in each class for the training sample and evaluated with the Hosmer-Lemenshow test on the test sample. [Chen et al., 2014] The risk classes were also evaluated by checking the stability of the risk classes' relative population. Figure 4.1 illustrates the model score distribution. The risk class thresholds will be concentrated on the lower scores as these have more occurrences.

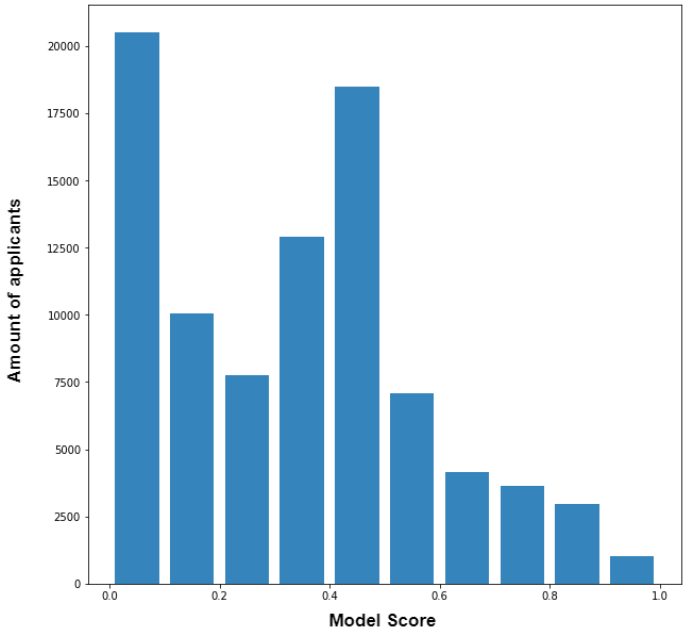

Figure 4.1: The scores from the network. Scores around 0 and 0.5 have more occurrences and the risk class segmentation will be finer in these regions.

4.6 White-Boxing with LIME

To deal with the problem of model acceptance, which is a critical part in real world use, the white-boxing method LIME was applied. The method exists as a python package created by Riberio et al. and has the ability to provide an explanation of one specific decision at a time. [Ribeiro et al., 2016] The method output is a score and a range for each input parameter that describes their importance for the model prediction. The sum of all input parameters' importance equals the model score and the number of input parameter combinations in the sample neighbourhood was set to 5000. The Euclidean distance function, denoted as D in chapter 3, was used with a kernel width of 0.75 times the number of input parameters. Explanations for applicants with different model scores and outcomes were created to get an understanding of how the model prioritize the input parameters and behaves in different scenarios. The method requires the same data as the model was trained on, in this case the normalized data. To display input parameters correctly, the output range given from LIME had to be unnormalized.

Chapter 5

Application Scorecard Evaluation

The models were evaluated with AUC-ROC on the test sample and compared to a Nordic bank's established application scorecard, scikit-learn's logistic regression and a relatively complex random forest model with 500 trees which was also implemented with scikit-learn. The logistic regression and the random forest were both trained on the same data as the network. Data set 1 and 2 consisted of credit card applications with a high and low limit respectively and approximately 125 000 and 825 000 data points were used. Data set 3 consisted of loan applications and approximately 75 000 data points were used. The three data sets were segmented as described in figure 4.5. Generally, the later test sample had a lower frequency of defaults than the training and validation samples. The model outputs were translated into default probabilities and compared to the observed defaults.

5.1 Data Set 1 - High Limit Credit Card Applications

5.1.1 Results

Data set 1 consisted of 125 000 data points with a default frequency of 2.5% and 66 different relevant inputs were found and used. Through the hyperparameter search described in chapter 4 the initial learning rate in the Adam gradient decent algorithm was set to be 0.00616 and the amount of nodes was set to 10 in a single hidden layer as this was found to provide sufficient performance. The node search is shown in figure 5.1. The AUC-ROC gain increases rapidly with the first few nodes but is almost constant after about 7 nodes. The only improvement detected with increased complexity after this point is the ROC-curve's shape. The amount of parameters (weights) in the model referred to as the chosen model, was 681.

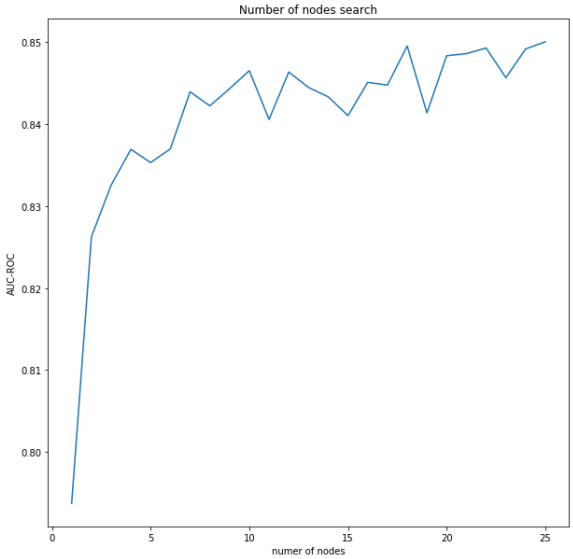

Figure 5.1: Search for the amount of nodes in an artificial neural network with one hidden layer on the validation set. The AUC-ROC improves a lot with the first few nodes and then the increase flattens out, disregarding the minor volatility.

Also, a more complex model was tested which had 700 nodes across 4 hidden layers, a total of 114 001 parameters, and resulted in the highest AUC-ROC, referred to as the complex model. Again, the model's amount of nodes were found in a hyperparamater search over a larger range. The ROC-curves for the training sample and the validation sample for the chosen and the complex model are shown in figure 5.2 and in figure 5.3 respectively. The ROC-curves for the test sample with the chosen and complex model are shown in figure 5.4.

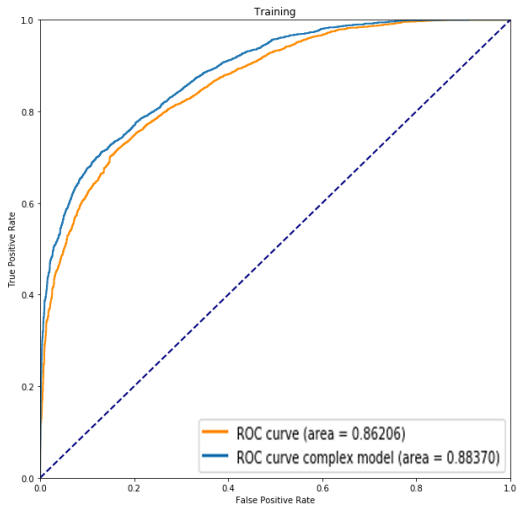

Figure 5.2: The ROC-curves for the training sample in data set 1 with AUC 0.8621 for the chosen model and AUC 0.8837 for the complex model. The chosen model is a network with 10 nodes in one hidden layer with a total of 681 parameters. The complex model is a network with 700 nodes across four hidden layers with a total of 114 001 parameters.

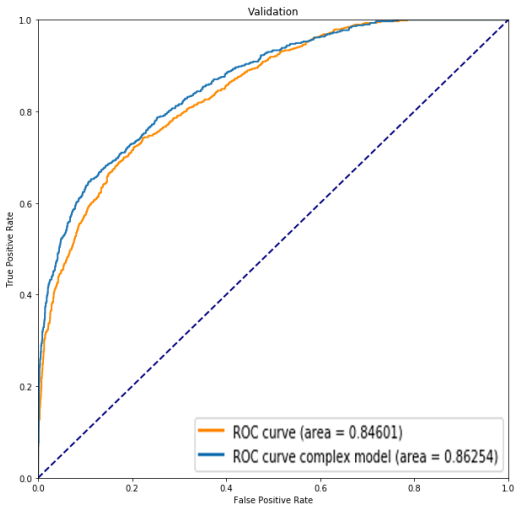

Figure 5.3: The ROC-curves for the validation sample in data set 1 with AUC 0.8460 for the chosen model and AUC 0.8625 for the complex model. The chosen model is a network with 10 nodes in one hidden layer with a total of 681 parameters. The complex model is a network with 700 nodes across four hidden layers with a total of 114 001 parameters.

The ROC-curves for the test sample for the chosen model, the bank's application scorecard and the logistic regression are shown in figure 5.5. The AUCs for the ROC-curves are summarized in table 5.1

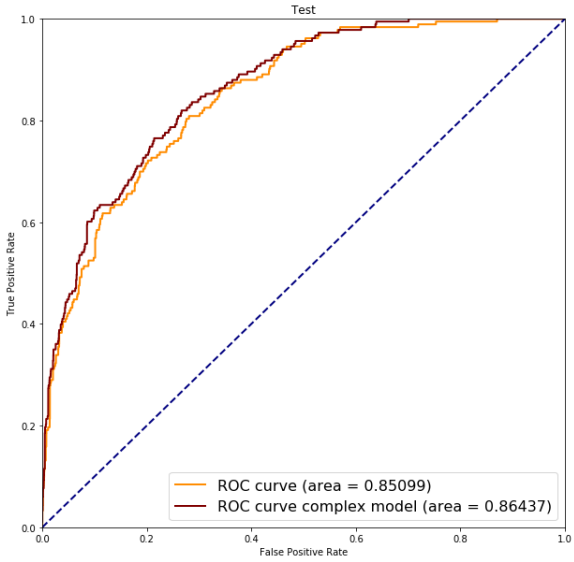

Figure 5.4: The ROC-curves for the test sample in data set 1 with AUC 0.8510 for the chosen model and AUC 0.8644 for the complex model. The chosen model is a network with 10 nodes in one hidden layer with a total of 681 parameters. The complex model is a network with 700 nodes across four hidden layers with a total of 114 001 parameters. Even with more than a 100 times the amount of parameters the performance increase with the complex model is negligible.

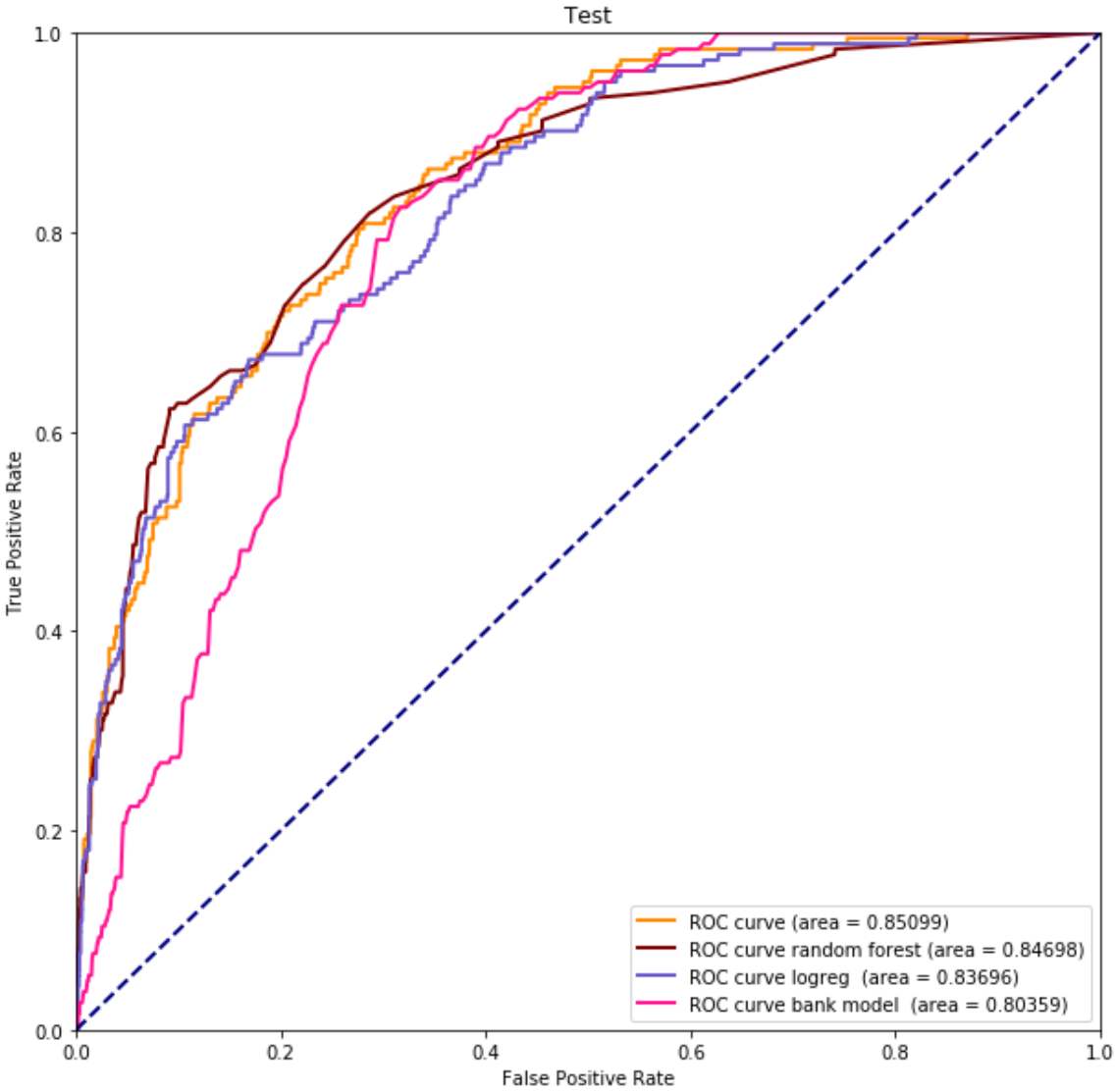

Figure 5.5: The ROC-curve for the test sample in data set 1 with AUC 0.8510 for the chosen network model with 10 nodes in one hidden layer, AUC 0.8036 for the Nordic bank's established application scorecard, AUC 0.8370 for the logistic regression and AUC 0.8470 for the random forest with 500 trees.

Table 5.1: Summarization of the AUC-ROCs on data set 1 for the chosen model, the complex model, the Bank's model, the logistic regression model and the random forest model with 500 trees. The chosen model is a network with 10 nodes in one hidden layer with a total of 681 parameters. The complex model is a network with 700 nodes across four hidden layers.

	Training AUC-ROC	**Validation AUC-ROC**	**Test AUC-ROC**
Chosen model	0.8621	0.8460	0.8510
Complex model	0.8837	0.8625	0.8644
Bank model	-	-	0.8036
LogReg	0.8352	0.8204	0.8370
Random forest	0.9999	0.8893	0.8470

The risk class segmentation for the chosen model resulted in table 5.2 and the Hosmer-Lemenshow test suggested that the expected default frequency, calculated from the training sample, had nearly zero percent probability of being the observed frequency for the test sample.

Table 5.2: The summarized risk class segmentation on the test sample for the chosen model, a network with 10 nodes in one hidden layer, on data set 1. The risk classes are sorted by the class thresholds and hopefully by probability of default. The population should optimally be the same in all risk classes if the data is stable and the expected PD is calculated with the training data.

Risk class	**Population**	**Observed PD (%)**	**Expected PD (%)**	**Class Thresholds**
1	1853	0	0.02	0.03
2	1390	0.07	0.02	0.04
3	1123	0.18	0.24	0.06
4	1093	0	0.30	0.09
5	991	0.10	0.49	0.13
6	917	0.44	0.71	0.19
7	923	0.98	1.38	0.27
8	931	0.65	1.14	0.33
9	1023	0.88	1.82	0.38
10	887	1.35	2.48	0.40
11	839	1.19	1.60	0.44
12	931	1.50	2.66	0.52
13	1011	3.07	5.31	0.66
14	1088	7.72	19.66	1

5.1.2 Analysis

5.1.2.1 Training and Validation Samples

Shown in figure 5.2, both ROC-curves for the training sample are smoothly concave and their AUCs are very similar. An increased model complexity did not provide enough performance gain to be worth the increased computing time and decrease in model acceptance. Over a hundred times more parameters resulted in only approximately a two percentage point AUC increase and no difference in the overall curve shape. Figure 5.3 illustrates the same outcome for the validation sample. The dropout method has clearly prevented overfitting since the performance on the two samples is very similar. Even regarding the overly complex model where overfitting is expected due to a lot of parameters which would allow the model to learn from the noise and reach a very high AUC on the training sample, no sign of this is shown. On the contrary, the random forest has clearly been overfitted to the training sample, as shown as the extremely high AUC in table 5.1

5.1.2.2 Test Sample

With figure 5.4 the conclusion was made that the chosen model is sufficiently good. The complex model has a slightly better shape and AUC but once again the differences are not great enough for the complex model to be preferable. Both models have worse performance on the test sample compared to the training sample but better compared to the validation sample. This is probably due to variance in the data set and in the default frequencies which makes the data harder to predict. However the curve shape in the validation sample is smoother than the test sample. The data points in the validation sample are similar to the training sample since they are from the same time period and this can explain why the validation ROC-curve is looking nicer than the out of time test sample ROC-curve. The model has learned features and overall generalization on the time period of the training and validation. The out of time test set does not share these generalized features completely.

The aim was to outperform the Nordic bank's established model and reach the same performance level as the state-of-the-art machine learning model random forest. This was successfully done as shown in figure 5.5 and in table 5.1. The bank's model was created before the time period of the data sets and thus did not perform very well on the test sample which is the most recent data. The development time period is a clear disadvantage compared to the ANNs which have been developed on more recent data. The main problem with the bank's model ROC-curve is the low slope in the left hand side which occurs when the model has issues with ranking the applicants. The ANNs, the random forest and the logistic model has less issues with these applicants. This is to be expected since the data is not constant over time and these models has the advantage of being trained on newer data. The convexity in the logistic regression ROC-curve indicates that the model is not complex enough. The chosen ANN model clearly outperforms both the bank's model and the logistic regression model. The chosen ANN model's ROC is similar to the random forest's ROC.

5.1.2.3 Risk Class Segmentation

Table 5.2 shows the result of the translation of the model output into a probability of default and risk class segmentation. The Hosmer-Lemenshow test seems to be too sensitive for the skewed data at hand and believes it has zero probability of being the correct segmentation. Even though the test result was bad the overall segmentation and prediction of defaults seems to be somewhat correct as the expected and observed PD both increases with the risk classes. Risk classes $7 - 11$ are not ranked properly regarding the PD and this issue might arise from the fact that the thresholds are closer together where the model has given many similar scores. It might help to merge these into one class to improve the classification. The population in the lower risk classes is larger than the ones in the higher ones. This indicated that the model has given more applicants a lower score in the test sample when compared to the training sample. This is reasonable since the overall default frequency is lower in the test sample. Because of the low PD and the relatively small test sample there was not any observed default in some of these classes. Overall the classes still underestimate the PD which might be because of external market or seasonality effects and can probably be corrected by utilizing macro-economic parameters to adjust the risk-classes and their expected PD.

5.2 Data Set 2 - Low Limit Credit Card Applications

5.2.1 Results

Data set 2 consisted of 825 000 data points with a default frequency of 1.0% and 17 different relevant inputs were found and used. Through the hyperparameter search described in chapter 4 the initial learning rate in the Adam gradient decent algorithm was set to be 0.00785 and the amount of nodes was set to 10 in a single hidden layer as this was found to provide sufficient performance. The amount of parameters (weights) in the model referred to as the chosen model, was 191. The ROC-curves for the training and the validation sample for the chosen model are shown in figures 5.1 and 5.2 respectively.

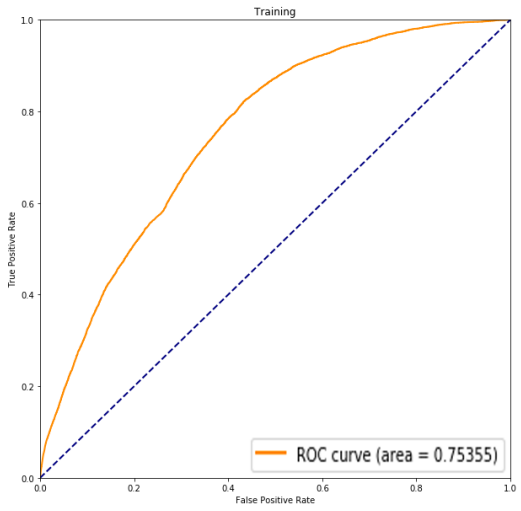

Figure 5.1: The ROC-curve for the training sample in data set 2 with AUC 0.7536 for the chosen model with 10 nodes in one hidden layer.

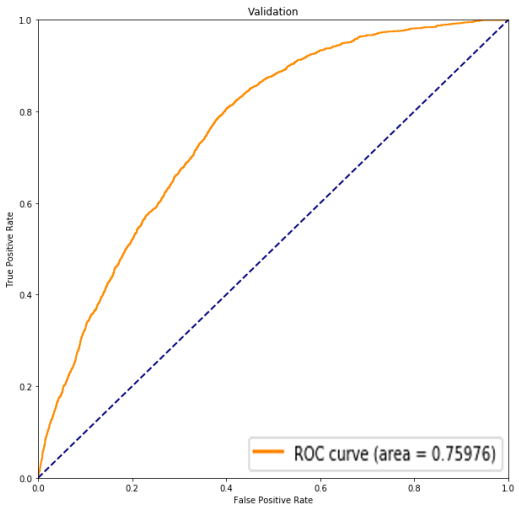

Figure 5.2: The ROC-curve for the validation sample in data set 2 with AUC 0.7600 for the chosen model with 10 nodes in one hidden layer.

The ROC-curves for the test sample with the chosen ANN model, the bank's application scorecard, the logistic regression and the random forest model are shown in figure 5.3. The AUCs for the ROC-curves are summarized in table 5.3

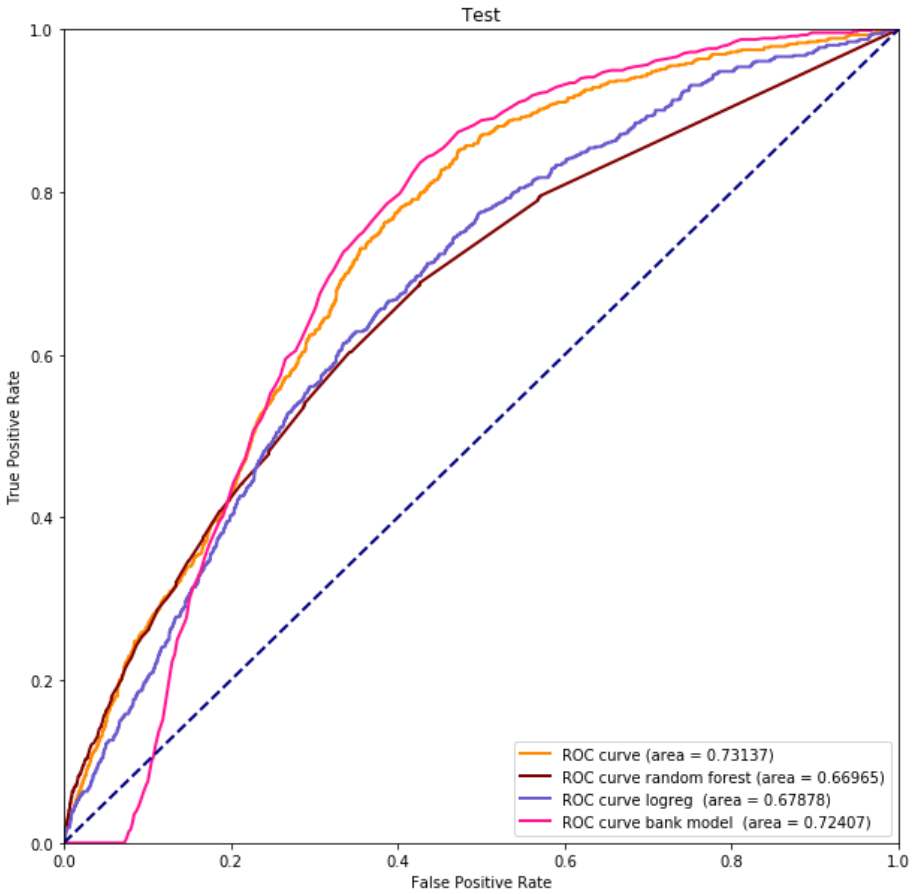

Figure 5.3: The ROC-curves for the test sample in data set 2 with AUC 0.7314 for the chosen ANN model with 10 nodes in one hidden layer, AUC 0.7241 for the Nordic bank's established application scorecard, AUC 0.6777 for the logistic regression model and AUC 0.6670 for the random forest model with 500 trees.

Table 5.3: Summarization of the AUC-ROCs on data set 2 for the network with 10 nodes in one layer, the Bank's model, the logistic regression model and the random forest model with 500 trees.

	Training AUC-ROC	**Validation AUC-ROC**	**Test AUC-ROC**
Chosen model	0.7536	0.7600	0.7314
Bank model	-	-	0.7241
LogReg	0.6949	0.6967	0.6788
Random forest	0.9923	0.7362	0.6670

The risk class segmentation for the chosen model resulted in table 5.4 and the Hosmer-Lemenshow test suggested that the expected default frequency, calculated from the training sample, had nearly zero percent probability of being the observed frequency for the test sample.

Table 5.4: The summarized risk class segmentation on the test sample for the chosen model, a network with 10 nodes in one hidden layer, on data set 2. The risk classes are sorted by the class thresholds and hopefully by probability of default. The population should optimally be the same in all risk classes if the data is stable and the expected PD is calculated with the training data.

Risk class	Population	Observed PD (%)	Expected PD (%)	Class Thresholds
1	12158	0.09	0.07	0.13
2	10134	0.07	0.09	0.17
3	9212	0.15	0.20	0.22
4	8925	0.16	0.27	0.27
5	8531	0.22	0.34	0.36
6	8524	0.25	0.43	0.41
7	8387	0.53	0.64	0.47
8	8044	0.55	0.95	0.53
9	8241	0.96	1.26	0.56
10	8427	0.93	1.53	0.578
11	8730	1.34	1.93	0.580
12	9147	0.87	1.34	0.60
13	6926	1.50	2.67	0.68
14	4614	1.76	3.86	1

5.2.2 Analysis

5.2.2.1 Training and Validation Samples

The ROC-curves on the training and validation sample in figures 5.1 and 5.2 illustrates nice concave shapes and no overfitting. This data set seems to be a lot tougher to model as shown by the relatively low AUC. The input parameters seems to have less predictive power when it comes to credit cards with low limit. This combined with the low default frequency limits how well this data set can be modelled even though there are a lot of data points. The AUC on the training and validation sample for the random forest model in table 5.3 indicates once again that this model is overfitted.

5.2.2.2 Test Sample

The test sample ROC-curve for the chosen model illustrated in figure 5.3 has the same smooth shape as the training and validation sample. The ROC-curve for the bank's model in the same figure displays a curious shape. In the left hand side, where the applicants with low scores are ranked, the model expect these applicants to default. But in this case the applicants with the lowest scores did not default.

This is not surprising since even the applicants with the absolute highest probability of default only had a 10% PD. The fact that the model only produces a low amount of unique scores and the difficulty in modelling amplifies these wrong rankings and this is shown as the curve going under the 45 degree slope which is the slope of a random ranking model. For the higher model scores, the bank model is surprisingly performing slightly better than the chosen ANN model. This is hard to explain since the bank scoring model development and inputs are confidential. Because of the parameters' less predictive power, both the logistic regression model and the random forest model have troubles with predicting this data set.

5.2.2.3 Risk Class Segmentation

Table 5.4 shows the same pattern as for data set 1 and once again it is the risk classes with thresholds close to each other that the issues arises from as well as the over time varying model score population.

5.3 Data Set 3 - Loan Applications

5.3.1 Results

Data set 3 consisted of 75 000 data points with a default frequency of 2.9% and 87 different relevant inputs were found and used. Through the hyperparameter search described in chapter 4 the initial learning rate in the Adam gradient decent algorithm was set to be 0.00359 and the amount of nodes was set to 10 in a single hidden layer as this was found to provide sufficient performance. The amount of parameters (weights) in the model referred to as the chosen model, was 891. The ROC-curves for the training and the validation set for the chosen model are shown in figures 5.1 and 5.2 respectively.

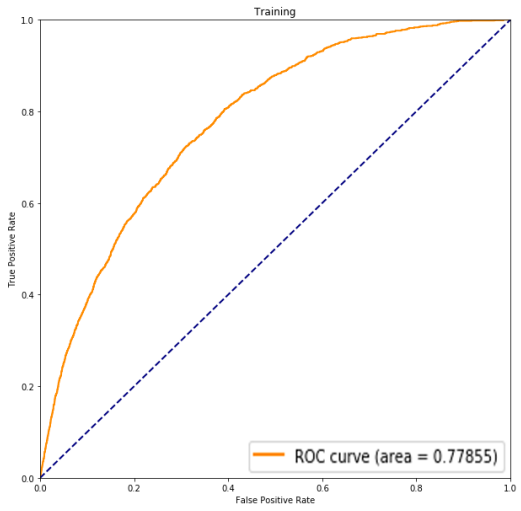

Figure 5.1: The ROC-curve for data set 3 for the training sample with AUC 0.7786 for the chosen model with 10 nodes in one hidden layer.

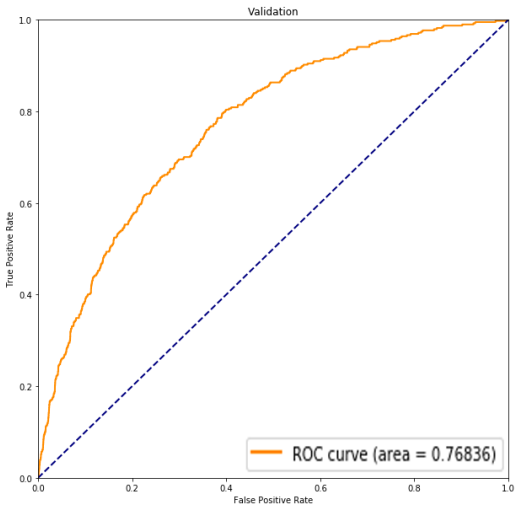

Figure 5.2: The ROC-curve for data set 3 for the validation sample with AUC 0.7684 for the chosen model with 10 nodes in one hidden layer.

The ROC-curves for the test sample with the chosen ANN model, the bank's application scorecard, the logistic regression and the random forest model are shown in figure 5.3. The AUCs for the ROC-curves are summarized in table 5.5

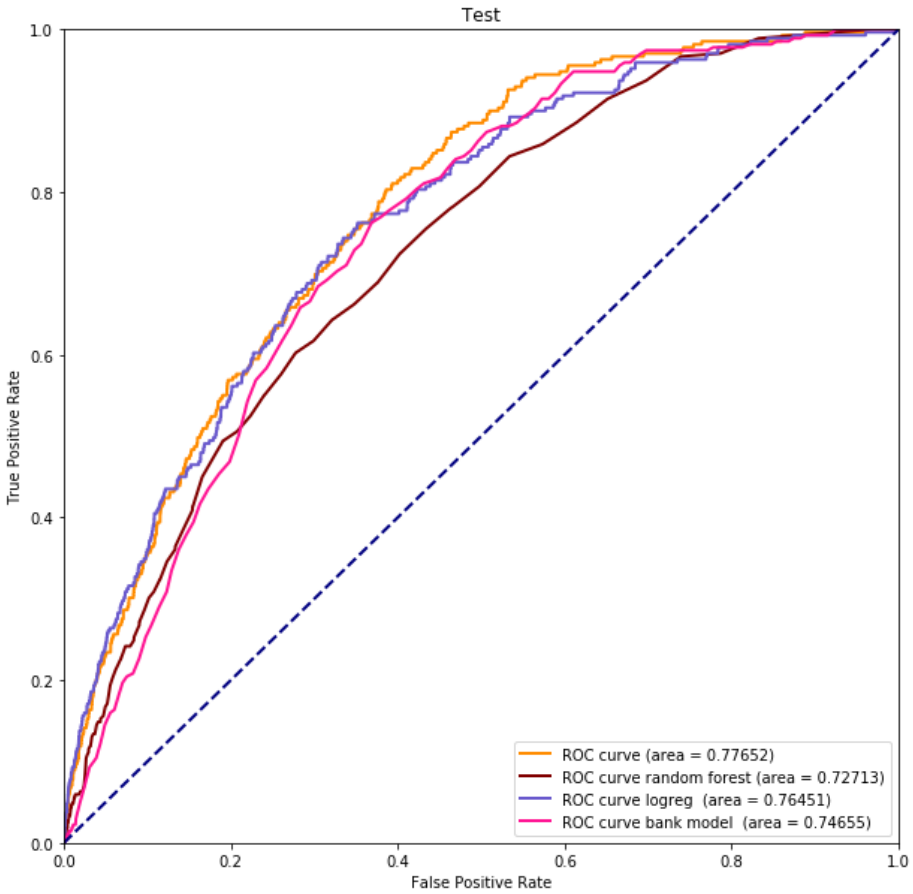

Figure 5.3: The ROC-curve for data set 3 for the test sample with AUC 0.7765 for the chosen ANN model with 10 nodes in one hidden layer, AUC 0.7466 for the Nordic bank's established application scorecard, AUC 0.7631 for the logistic regression model and AUC 0.7271 for the random forest model with 500 trees.

Table 5.5: Summarization of the AUC-ROCs on data set 3 for the network with 10 nodes in one layer, the Bank's model, the logistic regression model and the random forest model with 500 trees.

	Training AUC-ROC	**Validation AUC-ROC**	**Test AUC-ROC**
Chosen model	0.7786	0.7684	0.7765
Bank model	-	-	0.7466
LogReg	0.7483	0.7407	0.7645
Random forest	1.0000	0.7327	0.7271

The risk class segmentation for the chosen model resulted in table 5.6 and the Hosmer-Lemenshow test suggested that the expected default frequency, calculated from the training sample, had nearly zero percent probability of being the observed frequency for the test sample.

Table 5.6: The summarized risk class segmentation on the test sample for the chosen model, a network with 10 nodes in one hidden layer, on data set 3. The risk classes are sorted by the class thresholds and hopefully by probability of default. The population should optimally be the same in all risk classes if the data is stable and the expected PD is calculated with the training data.

Risk class	**Population**	**Observed PD (%)**	**Expected PD (%)**	**Class Thresholds**
1	1030	0.09	0.09	0.10
2	987	0.10	0.20	0.15
3	924	0.21	0.60	0.20
4	1055	0.38	0.66	0.27
5	961	0.1	1.28	0.34
6	1000	0.50	1.34	0.40
7	1041	0.96	1.85	0.45
8	1089	1.47	2.31	0.49
9	1119	1.52	3.10	0.52
10	1156	2.08	3.70	0.54
11	1141	2.45	4.76	0.58
12	1224	2.86	5.01	0.62
13	1209	4.38	6.89	0.69
14	1064	6.77	12.41	1

5.3.2 Analysis

5.3.2.1 Training and Validation Samples

The ROC-curves for training and validation in figures 5.1 and 5.2 shows no sign of overfitting and has a smooth concave curve shape. Loan applications seems to be harder to model than credit card applications with a high limit (data set 1) even though the input parameters were similar. One of the reasons can be that data set 3 has fewer data points.

5.3.2.2 Test Sample

The AUC-ROC for the test sample is greater than for the training and validation sample, shown in 5.5. This can be explained by the variance in the data. The bank's model ROC-curve illustrated in figure 5.3 has a lower slope in the left hand side due to the model development time period again was before the time period for the data samples. The chosen ANN model outperforms all the other models.

5.3.2.3 Risk Class Segmentation

Table 5.6 shows a better segmentation than for data sets 1 and 2. The population is more or less equal between the classes and both the observed and expected PDs increases nicely with the risk classes. Again, the observed PD is lower than the expected PD and might be solved by tuning the PD associated with the risk classes with macro-economic parameters.

5.4 Data Set 1 + 3

Due to the similarities between both the data set and the chosen ANN model for data set 1 and 3, these data sets were merged into one as an attempt to generalize the modelling process. Inputs from data set 3 that did not exist in data set 1 were excluded. When the two data sets were merged only data from the common time period was used.

5.4.1 Results

The resulting combined data set 1 + 3 consisted of 150 000 data points with a default frequency of 2.8% and 66 different inputs. The initial learning rate in the Adam gradient decent algorithm was set to be 0.00359 and the amount of nodes was found to provide sufficient performance with 10 nodes in one hidden layer. The amount of parameters (weights) in the model referred to as the combined model, was 681. The ROC-curves for the test sample with the combined ANN model, the bank's application scorecard, the logistic regression and the random forest are shown in figure 5.1. The AUCs for the ROC-curves are summarized in table 5.7

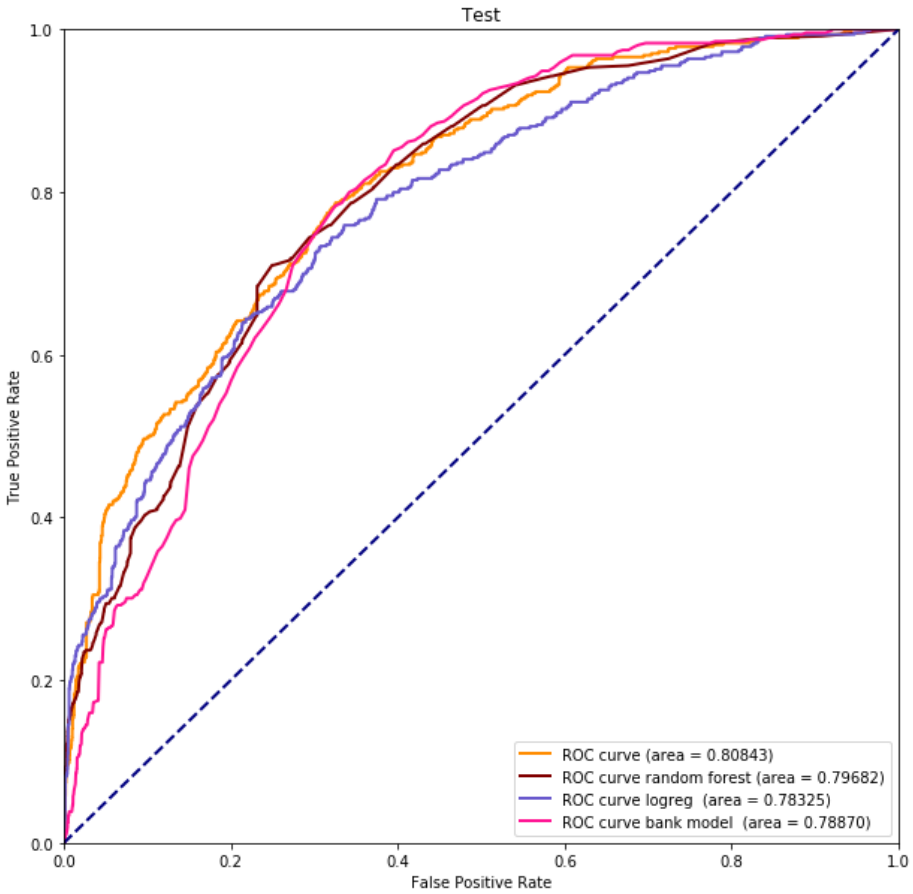

Figure 5.1: The ROC-curves for the test sample for data sets 1 + 3 with AUC 0.8084 for the combined ANN model with 10 nodes in one hidden layer, AUC 0.7887 for the Nordic bank's established application scorecard, AUC 0.7968 for the random forest with 500 trees and AUC 0.7833 for the logistic regression.

Table 5.7: Summarization of the AUC-ROCs on data set 1 + 3 for the network with 10 nodes in one layer, the Bank's model, the logistic regression model and the random forest model with 500 trees.

	Test AUC-ROC
Combined ANN model	0.8084
Bank model	0.7887
Combined LogReg	0.7833
Combined random forest	0.7968

The risk class segmentation for the combined ANN model resulted in table 5.8 and the Hosmer-Lemenshow test suggested that the expected default frequency, calculated from the training sample, had nearly zero percent probability of being the observed frequency for the test sample.

Table 5.8: The summarized risk class segmentation on the test sample for the chosen model, a network with 10 nodes in one hidden layer, on data set 1 + 3. The risk classes are sorted by the class thresholds and hopefully by probability of default. The population should optimally be the same in all risk classes if the data is stable and the expected PD is calculated with the training data.

Risk class	Population	Observed Freq.	Expected Freq.	Risk Class Thresholds
1	2174	0.14	0.19	0.15
2	1804	0.11	0.21	0.19
3	1622	0.31	0.37	0.25
4	1695	0.30	0.69	0.30
5	1744	0.23	0.95	0.35
6	1817	1.05	1.43	0.38
7	1711	0.82	1.69	0.40
8	1605	1.25	2.40	0.42
9	1564	1.09	2.29	0.44
10	1583	1.96	2.62	0.48
11	1919	2.40	3.84	0.53
12	1917	2.50	4.71	0.59
13	1861	2.69	5.87	0.69
14	1984	10.33	14.18	1

5.4.2 Analysis

5.4.2.1 Test Sample

The performance on the test data ROC-curve for the combined ANN model shown in figure 5.1 looks very similar to the earlier networks and the model had no problem with generalizing for both the credit cards with high limit and the loan application data sets. It managed to outperform the scoring made by the two separate bank models which is shown as a combined ROC-curve illustrated in the same figure. The combined ANN model also outperformed the combined logistic regression model and the combined random forest model.

5.4.2.2 Risk Class Segmentation

Table 5.8 suggests once again that the model overestimates the PD in the test sample. It is expected since the two data sets had this result separately. The combined data set seems to have a similar monotonously increasing scoring as data set 3.

Chapter 6

White-Boxing with LIME

White-boxing was only performed on data set 1 and the result is based on three chosen applicants. The applicants are chosen by their model score to showcase the model decision for as differing applicants as possible and gain maximum trusthworthiness. LIME provides an explanation for why an applicant has been assigned a specific model score by giving each input parameter an importance value.

6.1 Results

Figures 6.1 to 6.3 illustrates the input parameters' contribution to the model score for three applicants, one that defaulted and two that did not, all with different model scores. Only the twenty most important input parameters are shown in the figures and the values in parentheses are the applicants' actual input parameter values. The categorical features are represented with the values 1 or -1 in the figures. The input parameters with an underscore followed by a number is indicating parameters from different time periods with lower numbers being more recent. For the feature APPLIC_ROLE, 1 stands for the main applicant and -1 stands for co-applicant. For the feature GENDER_APPLIC, 1 stands for man and -1 stands for woman. For the feature APPL_TYPE_CODE, 1 stands for new credit application and -1 stands for increasing the limit on existing credit.

Negative input parameter scores contributes to a lower model score while positive input parameter scores contributes to a higher model score. A high model score indicates a higher probability of default. Therefore the negative contributions are colored green as these are parameters the model considers favorable for the applicant.

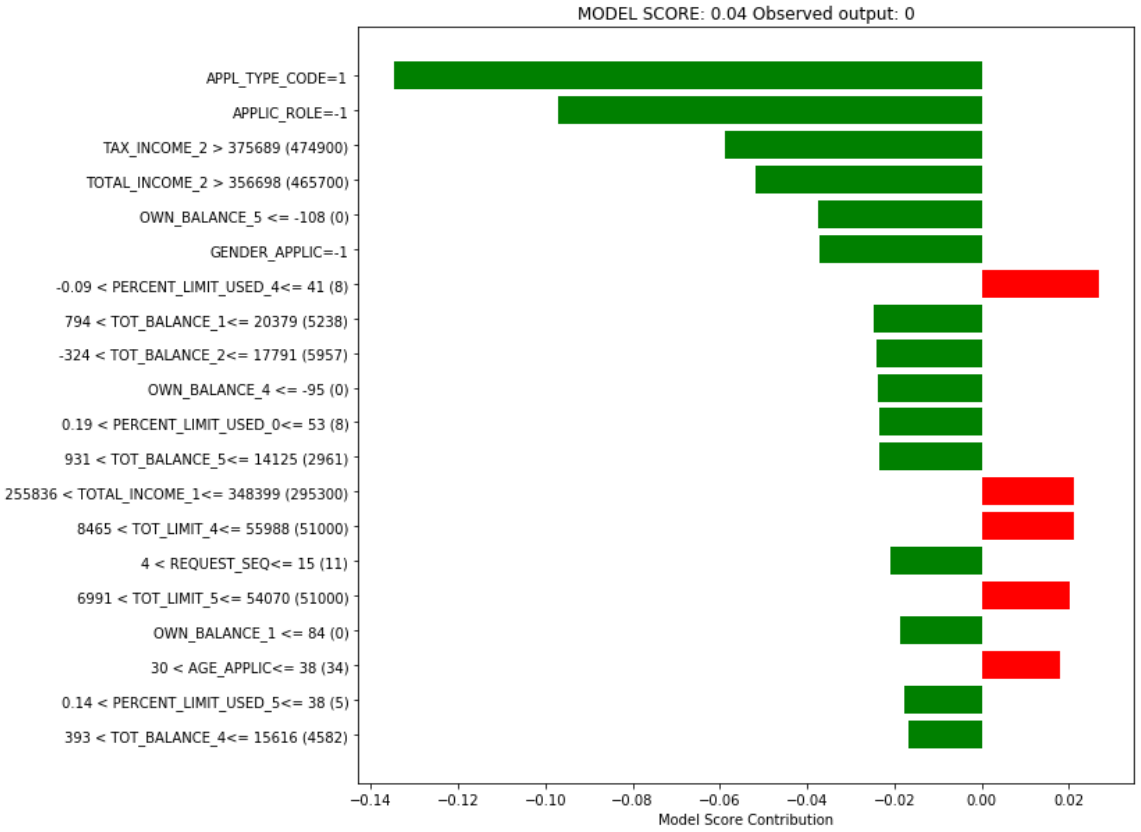

Figure 6.1: The model decisions for a non-defaulting applicant with a model score of 0.04. Illustrated are the twenty most important input parameters and the applicant's actual input parameter values are in parentheses. The input parameter scores are summed up to the total model score.

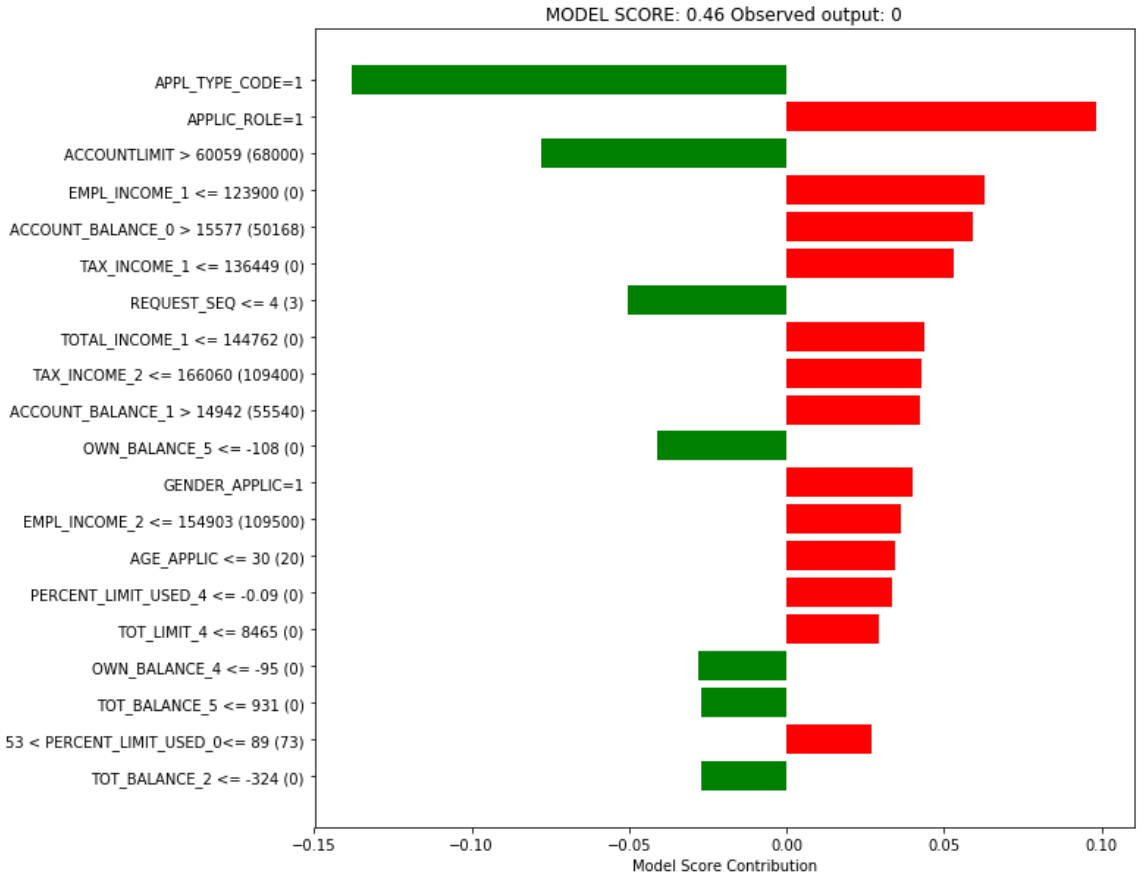

Figure 6.2: The model decisions for a non-defaulting applicant with a model score of 0.46. Illustrated are the twenty most important input parameters and the applicant's actual input parameter values are in parentheses. The input parameter scores are summed up to the total model score.

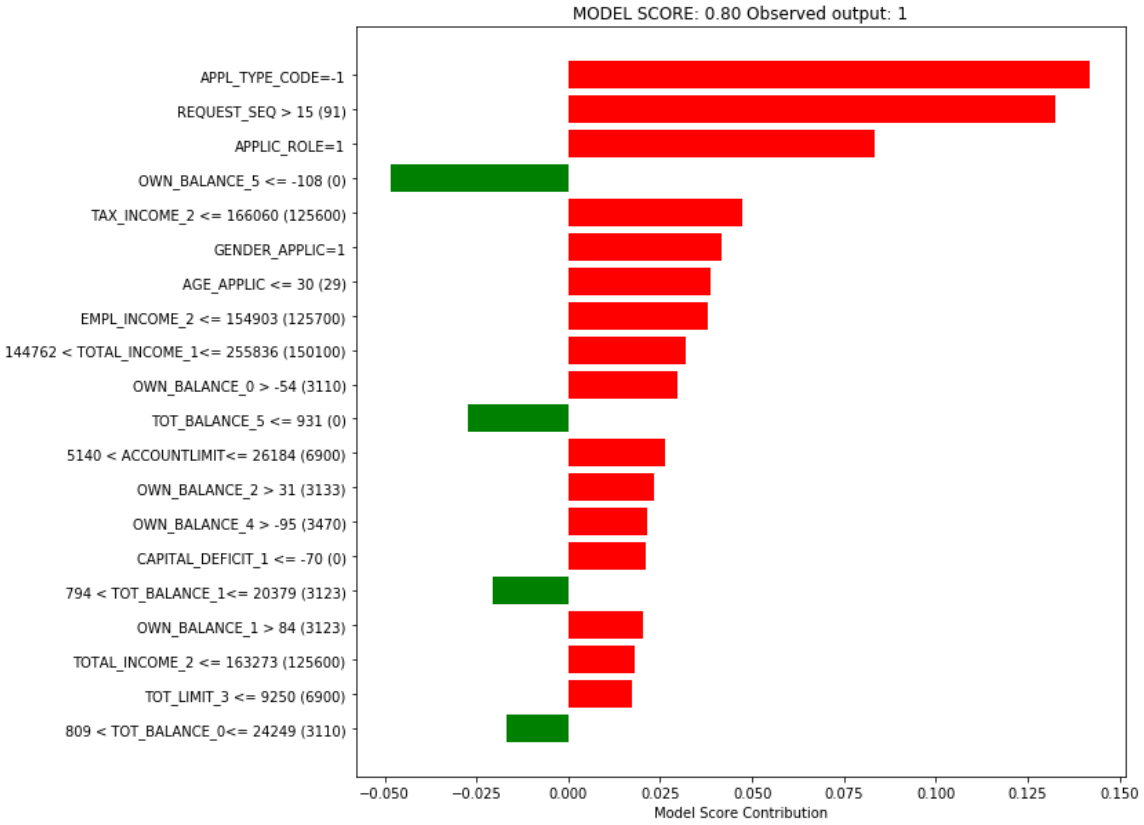

Figure 6.3: The model decisions for a defaulting applicant with a model score of 0.80. Illustrated are the twenty most important input parameters and the applicant's actual input parameter values are in parentheses. The input parameter scores are summed up to the total model score.

6.2 Analysis

The model seems to make consistent and logical decisions as the same features appear to have the similar impact in every decision even for the very different applicants shown in the figures. Some parameters, for example OWN_BALANCE, CAPITAL_DEFICIT and PERCENT_LIMIT_USED have inaccurate ranges due to the rounding LIME does when displaying them. This rounding is applied before the unnormalization which makes it impossible to accurately display the unnormalized values. The input parameters regarding the financial situation of the applicant, such as TAX_INCOME and EMPL_INCOME are ranked as important in all cases. Depending on whether you have a lot of money or not, correlates with the ability of paying back the credit. The parameters which implies that the applicant have a lot of money will contribute towards a non-defaulting decision. APPL_TYPE_CODE and APPLIC_ROLE, that is new credit or increased limit on existing credit and main- or co-applicant, are always ranked among the top contributors.

Having a co-applicant and applying for a new credit is deemed safer than being a lone applicant and applying for increased limit which seems logical.

In figure 6.3 REQUEST_SEQ contributes with a high positive score as the applicant had made 91 requests to credit institutions. For the applicants described in figures 6.1 and 6.2 this parameter has a low negative score, since they had only made a few requests. Finally, according to the model the input parameter GENDER_APPLIC, that is the gender of the applicant, also has a significant impact. The model seems to have decided that being a man implies a greater credit risk. Information about gender is of course not allowed to be used in modelling in practice as it is discriminatory and thus the modelling was also done without this parameter. When excluding gender the model performance was not notably different. A feature that one might think will affect the scoring is the time of the credit application, if the application is made right before paydays or a week or two before Christmas could indicate that the applicant in in an unstable economic situation. This theory was later tested by including the time of the application made as year, month and day as inputs to the network but very little significance was detected.

The logical, as determined heuristically, and the consistent results supply a very solid foundation for believing in the network. LIME provides a simple and easily understandable explanation for the model decision. This is both useful for the credit institution's implementation of the network and for customers requiring explanations for why they are scored the way they are.

Chapter 7

Conclusions

7.1 Performance

This work has shown that the artificial neural networks can provide good predictive power for application scorecards compared to current models and well known machine learning methods. The development process was made easier by the availability of data, as no extra data gathering or data manipulations were needed. During the network development the chosen model was not the best performing one, but rather the model was chosen based on the trade-off between performance increase, computing time and model transparency. If the main matter of interest is to maximize the predictive power the network can be pushed to extreme complexity to improve the performance marginally. In some cases it might be beneficial to construct a hybrid model with both the logistic regression used in the bank and the network such as in data set 2 where, illustrated in figure 5.3, the bank model is after a certain point outperforming the network. The same network architecture of 10 nodes in one hidden layer performed near optimal for all data sets and could even handle a combination of two different data sets. This suggests that simple artificial neural networks are very generalizable and can be implemented fast and easy.

The data segmentation described in chapter 4 was constructed to ignore seasonal effects. The performance on the out of time test sample did not suffer from this as shown by the high AUC-ROC compared to the AUC-ROC on the validation sample and implies that the model is stable. On the other hand, the risk-class segmentation did suffer from some seasonal or external effect which is shown as the overestimation of the expected defaults. This issue can be solved by a better risk class segmentation which takes marco-economic parameters into account and there should be no problems with using such existing methods. Another area of overall model performance improvement is the treatment of the missing values. In this work, the simplest possible measure was taken; the missing values were replaced with the median. Two examples of more advanced methods are suggested in chapter 8.

7.2 Transparency

The decision explanation provided by the LIME-method was deemed to have significant explanatory power and the output definitively helps model understanding and trustworthiness. As an additional way of understanding the model it might help to extract the network weights, that is the weights between the nodes in the layers. It is helpful to think of the neural network as a collection of logistic regressions. In the case of 10 nodes in one hidden layer the mathematics are simply 10 logistic regressions in the hidden layer followed by one final logistic regression in the output layer. LIME provides nice explanations to a questioning applicant, and together with the network weights the credit institution should have plenty of insight to the model. However, it is not that simple to understand why the weights are what they are and the linear approximation given does not explain combined features of the input data that the network has found.

7.3 Usage

The main contribution of this thesis is that the networks are found to be reasonably transparent after applying a white-boxing method but they are probably still too complex to be allowed to be used as the main credit scoring model in a credit institution, it can instead supplement the existing models. Since the network is fast and easy to continuously update it can potentially be used as a benchmark model for internal use.

Chapter 8

Further Work

This thesis provides a baseline for implementing neural networks as credit scoring models. Deeper analysis and extended work can be done for increased performance and further applications in the entire credit granting process. Applying artificial neural networks to the missing value problem could enhance the data for better training and overall performance. Suitable network architectures could be a generative adversarial network or an auto-encoder network which have the ability to create and recreate data. One example of further applications is the risk-based pricing which is the final product of the risk class segmentation. Risk-based pricing implies that the risk assigned to each applicant determines the credit interest rate the applicant should pay. An applicant with a larger probability of default needs to compensate for the extra risk the credit institution takes by paying more. To further strengthen the contention why neural networks are applicable as a credit scoring model, a comparison of the money loss between current models and the neural networks could be performed. The money loss is the money not recovered from a credit because of incorrect model predictions.

Bibliography

[Basle Committee on Banking Supervision, 1998] Basle Committee on Banking Supervision (1998). *International convergence of capital measurement and capital standards*. Basle Committee on Banking Supervision.

[Blomgren and Vitestam, 2017] Blomgren, L. and Vitestam, H. (2017). Machine learning and its applications within insurance hit rates and credit risk modelling. Available at https://lup.lub.lu.se/student-papers/search/publication/8915361 (2018-03-10).

[Bolton et al., 2010] Bolton, C. et al. (2010). *Logistic regression and its application in credit scoring*. PhD thesis, University of Pretoria. Available at https://repository.up.ac.za/bitstream/handle/2263/27333/dissertation.pdf?sequence=1 (2018-02-15).

[Chen et al., 2014] Chen, J., Dhar, S., Duffy, D., Liu, Y., Moore, R. O., Morokoff, W., Pedneault, M., Pole, A., Qian, Y., Rumschitski, D., et al. (2014). New performance measures for credit risk models. Tech. rep., Standard & Poor's Rating Services following 2014 Workshop on Mathematical Problems in Industry, held at New Jersey Institute of Technology, Newark, NJ, USA.

[Davis and Goadrich, 2006] Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM.

[Freedman, 2009] Freedman, D. A. (2009). *Statistical models: theory and practice*. cambridge university press.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. Updated 2017-01-30.

[Missaoui, 2018] Missaoui, B. (2018). Lecture notes in statistics in retail finance, chapter 4: Selection bias and reject inference. Available at http://wwwf.imperial.ac.uk/ bm508/teaching/retailfinance/Lecture4.pdf (2018-02-21).

[Ohlsson, 2017] Ohlsson, M. (2017). *Lecture Notes on Introduction to Artificial Neural Networks and Deep Learning (FYTN14/EXTQ40)*. Computational Biology and Biological Physics, Department

of Astronomy and Theoretical Physics, Lund University.

[Ribeiro et al., 2016] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

[Ruder, 2016] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*. Updated 2017-06-15.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[StatSoft, 2013] StatSoft, I. (2013). Textbook, electronic statistics. *Tulsa, StatSoft*. Available at http://www.statsoft.com/Textbook (2018-02-01).

[The Pennsylvania State University, 2018] The Pennsylvania State University (2018). Distinction between outliers & high leverage observations. Available at https://onlinecourses.science.psu.edu/stat501/node/337 (2018-03-23).

[Thonabauer and Nösslinger, 2004] Thonabauer, G. and Nösslinger, B. (2004). *Guidelines on Credit Risk Management: Rating Models and Validation*. Oesterreichische Nationalbank (OeNB). Available at https://www.oenb.at/dam/jcr:1db13877-21a0-40f8-b46c-d8448f162794/rating_models_tcm16-22933.pdf (2018-03-09).