CHARLES UNIVERSITY

FACULTY OF SOCIAL SCIENCES

Institute of Economic Studies

Tomáš Turlík

Neural networks and tree-based credit scoring models

Bachelor thesis

Prague 2018

Author: Tomáš Turlík

Supervisor: doc. PhDr. Ladislav Krištoufek Ph.D.

Academic Year: 2017/2018

Bibliographic note

TURLÍK, Tomáš. *Neural networks and tree-based credit scoring models.* Prague 2018. 36 pp. Bachelor thesis (Bc.) Charles University, Faculty of Social Sciences, Institute of Economic Studies. Thesis supervisor doc.PhDr. Ladislav Krištoufek, Ph.D.

Abstract

The most basic task in credit scoring is to classify potential borrowers as "good" or "bad" based on the probability that they would default in the case they would be accepted. In this thesis we compare widely used logistic regression, neural networks and tree-based ensemble models. During the construction of neural network models we utilize recent techniques and advances in the field of deep learning, while for the tree-based models we use popular bagging, boosting and random forests ensembling algorithms. Performance of the models is measured by ROC AUC metric, which should provide better information value than average accuracy alone. Our results suggest small or even no difference between models, when in the best case scenario neural networks, boosted ensembles and stacked ensembles result in only approximately $1\% - 2\%$ larger ROC AUC value than logistic regression.

Keywords

credit scoring, neural networks, decision tree, bagging, boosting, random forest, ensemble, ROC curve

Abstrakt

Jednou z najzákladnejších úloh kreditného skóringu je klasifikácia potenciálnych klientov žiadajúcich o úver na "dobrých" alebo "zlých", na základe pravdedepobnosti, že by neboli schopní splácať úver v prípade, že by im bol odsúhlasený. V tejto práci porovnávame často používanú logistickú regresiu, neuronové siete a ensemble modely založené na stromových metódach. Pri konštrukcii neuronových sietí používame nové metódy a poznatky z oblasti hlbokého učenia, zatiaľčo v prípade stromov používame populárne ensemble algoritmy bagging, boosting a náhodné lesy. Modely porovnávame na základe ROC AUC miery, ktorá by mala poskytnúť väčšiu informačnú hodnotu ako len samotná presnosť. Výsledky naznačujú malý alebo takmer žiadny rozdiel medzi modelmi. V najlepšom prípade, dosahujú neuronové siete, boosted ensemble modely a zložené ensemble modely len približne o $1\% - 2\%$ väčšiu ROC AUC hodnotu ako logistická regresia.

Klíčová slova

kreditní skóring, neuronové sítě, rozhodovací strom, bagging, boosting, náhodný les, ensemble, ROC křivka

Declaration of Authorship

I hereby proclaim that I wrote my bachelor thesis on my own under the leadership of my supervisor and that the references include all resources and literature I have used.

I grant a permission to reproduce and to distribute copies of this thesis document in whole or in part.

Prague, 30th July 2018

Signature

Bachelor Thesis Proposal

Thesis will try to introduce various machine learning techniques to answer the question of whether in terms of loan default prediction, they can perform comparably or even better than normal linear regression models. Default predictions used in big institutions are usually exclusively modelled by logistic regression, therefore I will try to show that machine learning models can replace/be used together with this normal approach.

Data used for the thesis are from Lending Club from 2007-2017. Lending Club is the biggest peer-to-peer lending platform in the US. The dataset contains 300 thousand completed loans with 30 relevant variables. The dataset will be split into a randomly selected training subset and a smaller randomly selected testing subset. The models will be constructed using the training subset and subsequently run on the testing subset to compare the performance of the models. Selected machine learning models will involve primarily decision trees & random forests (James et al., An Introduction to Statistical Learning) and artificial neural networks (Murphy, Machine Learning: A Probabilistic Perspective).

The thesis will contain a theoretical and an empirical part. In the theoretical part I will firstly review the current state of machine learning usage in economics and secondly review machine learning techniques. In the empirical part I will use the data to create different models for predicting default. In the final chapter I will compare the results and conclude.

Core Bibliography

1. Athey, Susan, and Guido Imbens. "The State of Applied Econometrics-Causality and Policy Evaluation." arXiv Preprint arXiv:1607.00699, 2016.

2. Varian, Hal R. "Big Data: New Tricks for Econometrics." Journal of Economic Perspectives 28, no. 2 (May 2014): 3–28.

3. Krauss, Christopher, Xuan Anh Do, and Nicolas Huck. "Deep Neural

Networks, Gradient-Boosted Trees, Random Forests: Statistical Arbitrage on the S&P 500." European Journal of Operational Research 259, no. 2 (June 2017): 689–702.

4. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An Introduction to Statistical Learning. Vol. 103. Springer Texts in Statistics. New York, NY: Springer New York, 2013.

5. Murphy, Kevin P. Machine Learning: A Probabilistic Perspective. Adaptive Computation and Machine Learning Series. Cambridge, MA: MIT Press, 2012.

Contents

1 Introduction

One of the functions of lending institution is primarily to act as intermediary for borrowers and lenders. As relatively small reward is shadowed by potential large loss stemming from borrowers defaults, early detection of problematic loans is one of the most important goals of lending institutions. One of the models used to fulfill this task, is the model, that can reliably predict probability of borrower's inability to repay the loan (defaulting), which can be referred to as credit scoring.

Historic overview of bankruptcy prediction studies by Bellovary et al. (2007) [1] provides information about trends in model usage in scientific studies since 1930s. Until Altman in 1968 published study using multivariate discriminant analysis, all studies were using simple ratios of economic indicators to evaluate companies. After 1968, discriminant analysis models were primarily used until 1980s, when logistic and probit models became more popular. Studies using neural networks came in late 1980s however they didn't overtook logistic/probit models usage until 1990s.

Louzada et al. (2016) [2] provide natural extension of this study by reviewing classification models for credit scoring used between years 1992 and 2015. Their findings reveal that most frequently used models in this period were neural networks. Second most popular were hybrid models which combine techniques from multiple models together (for example models which perform feature selection and optimization simultaneously). Other popular models include ensembles, support vector machines and logistic regression. Result wise, majority of tested methods were very similar with fuzzy logic systems and support vector machines being overall slightly better.

Neural networks have been used for a long time, however the largest advances in these models came in the last 10 years from the fields of deep learning and AI research. Main points of interest for these fields are "big data" projects involving image processing, natural language processing, speech and audio recognition and associated projects like self-driving cars and AI assistants. Even though it is not their primary objective, we will try to explore

the potential of these new techniques on a simple classification problem.

Relatively new methods like neural networks, support vector machines etc. are developed to match or even outperform simpler older models like linear discriminant analysis, logistic regression or decision trees. Our next goal will be to construct tree-based models to show that even this simple method, when used in conjunction with ensembling can match the more complex methods. Ensembling is a process in which group of models is used to reach final prediction. Twala (2009) [3] measures performance of ensembles of models on credit datasets with added artificial noise. Overall ensembles outperform any single model trained on datasets with or without added noise. Naive Bayes classifier, logistic regression and decision trees performed the best, while worst performance was achieved by k-nearest neighbors and neural networks. Similar results, where ensembles of models beat any single model were presented by multiple studies [4] [5].

Majority of literature which compares models credit scoring models, measures model performance by classification accuracy and tries to maximize this statistic. We argue that because it is generally not known what are the costs associated with accepting bad loan applicant (false positive case) or declining good one (false negative case), only that certainly second case is preferred over the first one, we cannot use accuracy as the primary indicator of model performance. Instead we try to optimize statistic ROC AUC (Receiver Operator Characteristic Area under curve) which tries to capture the whole classification performance of a model without the need of choosing exact decision threshold.

2 Literature Overview

Large amount of literature has been published comparing credit scoring models. Here we present mostly scientific papers which review overall usage of methods, but also those with smaller scope which are of great relevancy to this thesis.

Henley and Hand (1996) [6] review statistical methods used in consumer credit scoring. They conclude that there is not an universally best method as this is deeply data specific problem but note that speed of classification and interpretability of model decisions are additional measures which should be considered when building a model. In this way logistic regression, k-nearest neighbors and some tree based methods are preferred to methods such as neural networks. On the other hand, the authors note that neural networks are better in the situations, when data structure is not well known as they can decide which features to use in which way. Simultaneously they argue that credit scoring has a considerable history with a solid understanding of data and therefore new classification models are unlikely to improve current models significantly. The point of possible improvement, in their opinion is in the collection or engineering of new features or in the fundamental change of the classification strategy.

West (2000) [7] compares traditional multilayer perceptron architecture (MLP) of neural network with four different architectures and commonly used models acting as a baseline. On the two used datasets mixture-of-experts and radial basis function models perform significantly better than MLP models but not as good as logistic regression, which is on average method with the lowest overall error rate. Non-parametric methods k-nearest neighbors, kernel density estimation and decision trees perform significantly worse than all previous methods, which authors partially attribute to the small size of the used datasets.

Baesans et al. (2003) [8] benchmark eight state of the art classification algorithms. Best obtained results are on average produced by support vector machines and neural networks, however they are only marginally better and

basic models like linear discriminant analysis and logistic regression are very comparable or even better in some specific cases. Authors conclusions are that this is because most credit scoring datasets are only weakly non-linear, thus potential gain from using neural networks is minimal.

Alaraj et al. (2014) [9] use simple neural networks (2 hidden layers, 20-40 neurons) to classify consumer loans candidates using ensembles of neural networks. They explore new techniques called CV-bagging and Bagging-CV. CV-bagging creates n bags on each partition which is created during k-fold cross validation process, creating in total nk neural networks. Bagging-CV does the same but vice versa, first creates n bags and then partitions each bag by k-fold cross validation process. Results suggest that these methods are comparable to the basic k-fold CV, but produce more stable models with smaller variance. However this can be partially explained by the vast number of models constructed in this methods compared to the k-fold cross-validation.

2.1 Internal rating models

Basel II in 2002 introduced internal ratings-based approach, which allows lending institutions to use their own internal credit risk models.

Belás and Cipovová (2007) [10] summarize guidelines for internal rating models, as well as provide description of the process of model development. Some obvious fundamental requirements include objectivity, sufficient accuracy and consistency between multiple models. Banks generally classify potential loans into multiple rating categories based on the risk instead of just binary categories, representing good and bad borrower, which are often used in scientific studies.

Process of internal rating model development involves generation of the dataset, development of the model itself, calibration, qualitative and at the end quantitative validation. Dataset generation is composed of data cleansing, dealing with missing values, investigating correlation between features and collection of enough data to cover whole target population. During cal-

ibration, outputted probabilities are associated with credit rating category. Validation is often performed by leave-n-out cross-validation, where models are trained on datasets without n samples and subsequently tested on those n samples (this occurs until all samples have been left out). Qualitative validation involves model design testing, data quality testing and internal test of model utilization. Quantitative validation is accomplished by back and stress testing of models. After model passes the whole process, it can be deployed, but periodical checking of its performance on new data is still required.

The Single Supervisory Mechanism, which comprises of European Central Bank and all central banks of Euro participating countries, provides extensive guidelines for models predicting probability of default [11]. Some noteworthy requirements include minimal list of features used in the model, namely client type, product type, region of client's location, past delinquencies and maturity of a loan. The model is required to have solid theoretical assumptions and has to consider possible change of the economic conditions. They also set requirements for rating categories, as well as procedures associated with their modification, in case of sudden change in economic conditions or unexpected increase of default rates.

Austrian central bank Oesterreichische Nationalbank (2004) [12] provides overview of most used practices linked with internal rating models in banks.

Internal rating models of lending institutions can be generally divided into *heuristic*, *statistical*, *causal* and *hybrid form* models.

Most notorious *heuristic* tools are questionnaires, which appoint given number of points to each question and the end, borrowers who cross some point threshold are deemed solvent.

Other tools include:

- Qualitative systems, similar to questionnaires, but rate each answer on a scale.
- Expert systems, which try to create complex rule based models essentially trying to emulate analytical behavior of real human credit experts.

- Fuzzy logic systems, which introduce fuzzy logic into the expert systems.

Basic expert systems are very similar to decision trees, but are not algorithmically created. Instead they are designed in advance by human experts to automate decision process. They use *knowledge base*, collection of various credit risk related research, as a source for series of rules to perform the final decision. More complex expert systems also incorporate statistical models [13].

Fuzzy logic systems act as an extension of expert system. Compared to the basic expert systems ratings are not based on some solid threshold but instead implement the concept of *partial truth* i.e. decision which rates a feature as good if it is higher than a, and bad if it is lower or equal, would in basic expert systems rate value $a + \epsilon$ as good where fuzzy logic system could rate it as 90% bad and 10% good, thus capturing the whole spectrum instead of just dichotomous decision.

Statistical models use past data to score borrowers in contrast of mostly subjective scoring in heuristic methods. Most frequently used "classical" models are Linear Discriminant Analysis (LDA) and Logistic (or probit) regression (LR). In practice logistic regression and probit are very similar and LR is mostly preferred just because of the easier interpretability.

In comparison to LR, LDA requires normality of independent variables and homoscedasticity, assumptions which can be rarely fulfilled. Even if the LDA assumptions hold, it has been shown that with large sample size differences in performance between LDA and logistic regression are minimal [14].

Logistic regression has been also shown to be the most common model used in the lending institutions [15] [16], partially because of its speed and easy interpretability. Because of these reasons, we will use logistic regression as the baseline model in our experiments.

Causal models use financial theory as a key factor in decision making. Most famous causal models utilize theory from option pricing or cash flow

models.

Option pricing model, used predominantly for firms, assumes that a default will happen if the borrower's debt exceeds the economic value of the his assets. In this model, loan taken by a firm is considered as an option which in the case of default awards the firm itself to the lender. At this point standard option pricing theory is used to calculate the price and probability of the option's exercise (which corresponds to the probability of default).

Cash flow models are used to predict future cash flows therefore indirectly probability of default. Cash flows are the sole consideration in this model, and other borrower's characteristics are not considered. Option pricing models often utilize cash flow models to impute economic value of the firm.

Hybrid form of models is the method which combines previous mentioned models together in order to improve classification performance. Hybrid models can be generally divided into two categories.

Horizontal linking combines models which are good with quantitative data (statistical, causal) and models which are good at qualitative data (heuristic). Final assessment is then function of both models' outputs.

Vertical linking is characterized by its ability to implement models which can override decision made from models which are higher in the hierarchy. For example, credit analyst can modify final classification of the statistical or causal model on his own discretion. Specification of the exact rules in which conditions can the classification be modified are strongly needed to prevent moral hazard and other problems.

3 Models

Because lending institutions often gradually rate applicants, higher grade representing higher risk and higher interest rate, it is possible even for highly risky loans to be profitable if the interest rate is sufficiently high. Construction of this kind of classification system is however more complex and requires further data specific to the each institution.

Our ambition is simpler and more general, that is to categorize loan applicants into two groups, group which will likely repay the whole loan without problems and group which will likely be not able to repay. All models output probability of default and optimal threshold determines the class. We will treat defaults as a positive case and fully paid loans as a negative.

3.1 Logistic regression

Logistic regression is a binary response model, where output of the regression represents probability of successful event (in our case default). LR can be seen as a linear model where dependent variable is a logit transformation of y (also called log odds), $\mathbf{x}$ is matrix of independent variables (with first column filled with 1 to allow for intercept) and $\boldsymbol{\beta}$ is a vector of coefficients.

$$\log(\frac{P(y=1|\boldsymbol{x})}{1-P(y=1|\boldsymbol{x})}) = \boldsymbol{x}\boldsymbol{\beta} \tag{1}$$

Probability $P(y=1|\boldsymbol{x})$ is then,

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{x}\boldsymbol{\beta}}} \tag{2}$$

Maximum likelihood estimation is the most often used method to estimate coefficients of logistic regression. To illustrate the process we present simple algorithm to find optimal $\boldsymbol{\beta}$. Let $P(\boldsymbol{x}_i, \boldsymbol{\beta})$ represent $P(y=1|\boldsymbol{x})$ for data point i from the logistic regression with parameters $\boldsymbol{\beta}$. To simplify calculations we will work with log-likelihood:

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{N} y_i \log[P(\boldsymbol{x}_i,\boldsymbol{\beta})] + (1-y_i)\log[1-P(\boldsymbol{x}_i,\boldsymbol{\beta})] = \sum_{i=1}^{N} y_i \boldsymbol{\beta}^T \boldsymbol{x}_i - \log[1+e^{\boldsymbol{\beta}\boldsymbol{x}_i}] \tag{3}$$

This function is strictly concave therefore to find global maximum, we have to solve

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} \boldsymbol{x}_i [y_i - P(\boldsymbol{x}_i, \boldsymbol{\beta})] = 0 \tag{4}$$

which can be approximately solved by Newton's iterative method

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} - \left[\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T}\right]^{-1}_{\boldsymbol{\beta}=\boldsymbol{\beta}_{t-1}} \left[\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right]_{\boldsymbol{\beta}=\boldsymbol{\beta}_{t-1}} \tag{5}$$

In practice this algorithm is slow and does not always guarantee convergence. Therefore more complex algorithms are used in practice, for example Newton CG (Newton Conjugate Gradient), SAG (Stochastic Average Gradient) or SAGA.

3.1.1 Regularization

Main idea behind regularization is to introduce bias by controlling parameter values, but at the same time reduce variance with the goal of reducing overall generalization error. This works because models with low bias usually tend to have larger variance and vice versa.

L1 (Lasso) and L2 (Ridge) norm regularization work by penalizing models with large coefficients. L1 regularization pushes weights toward zero, thus also doing feature selection at the same time, while L2 pushes weights toward small values near zero (both methods require standardized data). Parameter λ is called shrinkage parameter and influences size of the regularization, with higher values pushing coefficient closer to 0. In the case of L2 regularization we want to find $\boldsymbol{\beta}$ which leads into

$$\max_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{N} \left[y_i \boldsymbol{\beta}^T \boldsymbol{x}_i - \log[1 + e^{\boldsymbol{\beta} \boldsymbol{x}_i}] \right] - \lambda \frac{1}{2} \boldsymbol{\beta}^T \boldsymbol{\beta} \right\} \tag{6}$$

and for L1 regularization

$$\max_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{N} \left[y_i \boldsymbol{\beta}^T \boldsymbol{x}_i - \log[1 + e^{\boldsymbol{\beta} \boldsymbol{x}_i}] \right] - \lambda ||\boldsymbol{\beta}|| \right\} \tag{7}$$

Solutions to these equations can again be approximated by the iterative methods, value of λ which minimizes loss function is chosen.

3.2 Artificial Neural Networks

Artificial neural networks (or just neural network) try to emulate brain neurons and links between them. Every neuron has some function and the corresponding strength of the connection between neurons denotes importance of given neuron in the model.

Neural networks have existed for a long time, from the "electronic brain" in 1943, perceptron in 1957 and later multi-layered perceptron in 1986. However their performance was worse then widely used regressions or support vector machines, which was primarily caused by inadequacy of computational power, unavailability of large datasets and lack of advanced techniques used in the learning process.

In 1989 Cybenko [17] proved that neural networks with just one hidden layer and finite number of neurons can approximate any continuous function of n variables on compact subset of $\mathbb{R}^n$ if it is using sigmoid activation function. Later Hornik et al. [18] [19] extended the theorem to work with any sufficiently smooth activation function, however there still wasn't efficient way to find the parameters of the network. This theorem is commonly referred to as the Universal Aproximation Theorem.

At the beginning of the new millennium models like support vector machines were almost universally better than neural networks. This state continued until the increase in computational power and new discoveries in the field caused deep neural networks to slowly beat other methods in various fields.

Name "neural network" itself can refer to general architecture of nodes with connections, however it is often used as a shorter term for "multilayer perceptron" or "feedforward artificial neural network" (which usually refer to the same architecture).

The goal of a neural network (NN) is to use feature vectors $\mathbf{x}$ to find a function $f^*(\boldsymbol{x})$ which approximates unknown intrinsic function, mapping $\mathbf{x}$

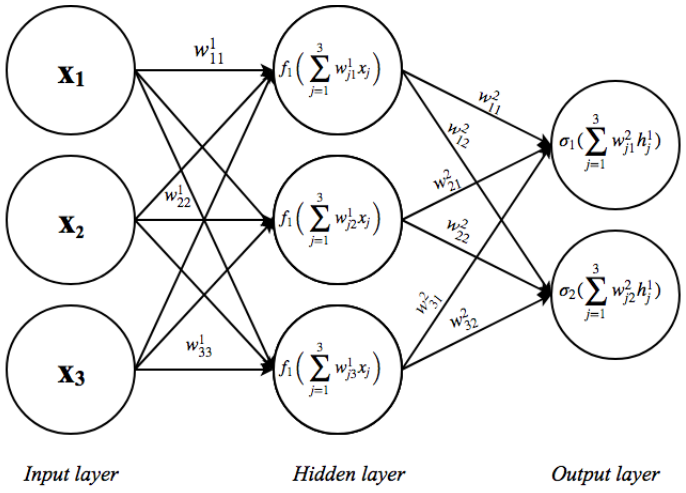

Figure 1: Simple neural network with 1 hidden layer
Every neuron in the previous layer is connected with the neuron in the next layer, strength of this connection is expressed by the weight. Some weights are not annotated for clarity.

to y, such that expected value of a loss function is minimized.

$$f^*(\boldsymbol{x}) = \underset{F(\boldsymbol{x})}{\operatorname{argmin}} E_{y,\boldsymbol{x}} L(F(\boldsymbol{x}), y) \tag{8}$$

Compared to the most used methods, NN do not assume any functional form of the intrinsic function and can in theory model any function.

Figure 1 shows simple neural network classifier with one hidden layer. Input layer h_0, with neurons representing elements of vector of features $\mathbf{x}$, is connected by each neuron to the hidden layer. In this layer function f_1, called activation function, is applied to a weighted sum of inputs using weights w^1. In the next step, we treat this hidden layer as a new input layer h_1 connected to the output layer.

Network is classifying into 2 categories as output layer uses softmax activation function, which is an extension of the sigmoid function used for more than 2 classes. The same result could be achieved by just one node if sigmoid function was used, but this way we can easily augment output layer in case we have more than 2 classes. Value of each node in the output layer represents probability of belonging to the one of the classes. During training, weights $\boldsymbol{w}$ are modified to better approximate intrinsic function $f^*(\boldsymbol{x})$.

Neural network with more than 2 hidden layers is generally called deep

and its field is called *deep learning*. Even though Universal Aproximation Theorem states that one hidden layer is sufficient to approximate any reasonable function, this layer can be unattainably large and similar results can be achieved faster and more efficient by using more layers with lower number of nodes.

Because neural networks are often build with intentions of processing large amounts of data, we split dataset into *mini-batches* and train the network one batch at the time until the whole dataset passes through the network, which is called an *epoch*. Training involves running as many epochs as is required for no further improvement in performance.

Overview of ANN training

1. *Forward Propagation*: Inputs are passed through hidden layers with corresponding *activation function*, outputs are used as input into another hidden layer until we reach output layer which outputs class provabilities and the loss is calculated by the *loss* function.
2. *Backpropagation*: *Optimizers* compute gradient of the *loss function* and adjust parameters in order to reduce loss.
3. During training, steps 1 and 2 are repeated and performance on the validation split is periodically checked, until no improvements occur.
4. Model is modified, trained and compared by validation performance to the previous models until the best model is found.

3.2.1 Forward Propagation

Before the first run of data through the network, weights are initialized randomly from a interval characteristic to each NN (dependent on architecture, activation functions etc.) [20] [21]. Right weight initialization is crucial, as wrong initial values can completely disrupt the learning process.

During the first run, weighted sum of nodes in a layer is passed into activation function f of each node in the next layer.

This can generally be expressed by

$$h_i = f_i(\sum_j w^i_{jk} a_j + b_i), \tag{9}$$

where w^i_{jk} represents the weight going from j-th node in h_{i-1} layer into k-th node in h_i layer, a_j represent output values of the previous layer h_{i-1} and b_i is an optional bias term, which is the same for each layer and can be compared to the intercept term in linear regressions which. We refer to the set of all trainable weights and biases as the parameters $\boldsymbol{\theta}$ of NN.

Activation functions

Activation functions are generally non-linear monotonic differentiable functions. Formulas 10a - 10d provide currently most used activation functions and their derivatives.

Sigmoid and hyperbolic tangent are among the most popular activation functions, however they can suffer from saturating non-linearities/vanishing gradients problem.

The problem can be illustrated with sigmoid activation function. If we stack multiple sigmoid layers of a single neuron, with weights set to 1 and biases set to 0, and send 0 through it, then $\sigma(0) = 0.5 \rightarrow \sigma(0.5) = 0.62 \rightarrow \sigma(0.62) > 0.62$... until we reach values very close to 1. Now, during back-propagation, derivative of sigmoid for values near 1 is going to 0 and therefore learning is very slow or even non existent.

Partially because of this problem, Glorot et al. in 2011 [22] came up with the activation function called rectified linear unit (ReLU) and also showed that it has generally better performance than sigmoid or hyperbolic tangent. There can be a similar problem as in a sigmoid/tanh case, called *dying ReLU*, where if weights connecting to one neuron become very negative then during training it can't be possible to change them as all gradients passing through this neuron are zero. Because of this there are many variations of ReLU, one of them being Leaky ReLU [23] that replaces 0 for negative inputs by

small multiple of the input.

$$\sigma(x) = \frac{e^x}{e^x + 1} \qquad \frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x)) \tag{10a}$$

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \qquad \frac{d\,tanh(x)}{dx} = 1 - tanh^2(x) \tag{10b}$$

$$ReLU(x) = max(0, x) \qquad \frac{d\,ReLU(x)}{dx} = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases} \tag{10c}$$

$$LReLU(x) = max(0.1x, x) \qquad \frac{d\,LReLU(x)}{dx} = \begin{cases} 0.1 & x \leq 0 \\ 1 & x > 0 \end{cases} \tag{10d}$$

Loss functions

After running data through the network during the training phase, predicted probabilities and actual labels are used to compute the loss. Loss function $L(\hat{y}, y)$, where y in our case represents label 0 or 1 (non-default/default) and $\hat{y}$ is the probability of default, is generally required to be continuous and differentiable on its domain as we need to calculate gradient during the backpropagation. One of the universal loss functions used in the machine learning is the mean square error.

$$MSE(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{n} \sum_i (\hat{y}_i - y_i)^2 \tag{11}$$

Derivative of this loss function for one observation i, given that $\hat{y}_i = \sigma(G_i)$, where G_i represents weighted sum of inputs from the previous layers with a bias term, is

$$\frac{\partial MSE(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\partial G_i} = 2(\sigma(G_i) - y_i)\sigma(G_i)(1 - \sigma(G_i)) \tag{12}$$

However, as we can see when the $\hat{y}_i$ gets closer to the value of the label, derivative becomes very small and the learning process slows down. This is one of the reasons why it is not often used in the deep learning.

Loss functions used nowadays came from information theory, few popular

of them are negative log-likelihood and cross-entropy.

$$\text{Binary cross-entropy loss} = H(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_i \Big[y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i) \Big] \tag{13}$$

if we compare its derivative to the MSE function,

$$\frac{\partial H(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\partial G_i} = -y_i(1 - \sigma(G_i)) + (1 - y_i)\sigma(G_i) = \sigma(G_i) - y_i \tag{14}$$

we can see that it is esentially the same expression as in the case of MSE derivative, just without $2\sigma(G_i)(1 - \sigma(G_i))$, which helps to solve the issue of slow learning.

3.2.2 Backpropagation

Backpropagation is the most important step in the NN training, it's the point when the networks actually "learn". Let output of our model be called $p(\boldsymbol{x}; \boldsymbol{\theta}) = \hat{\boldsymbol{y}}$, where x is input and $\boldsymbol{\theta}$ are parameters of our model. We try to find $\boldsymbol{\theta}^*$ such that the cost function $J(\boldsymbol{\theta})$ which represents sum of all losses from our data, is minimized.

$$\boldsymbol{\theta}^* = \underset{\theta}{\operatorname{argmin}}\, J(\boldsymbol{\theta}) = \underset{\theta}{\operatorname{argmin}} \sum_i L(p(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i) \tag{15}$$

The process of finding the optimum in NN is called the *gradient descent*. First, we calculate $\nabla_\theta J(\boldsymbol{\theta})$, to get the direction of the steepest increase, and then we update $\boldsymbol{\theta}$ to $\boldsymbol{\theta} - \alpha \nabla_\theta J(\boldsymbol{\theta})$ to go directly against the steepest increase. Parameter α is called learning rate and corresponds to how large steps do we want to take.

This gradient always exists because cross entropy loss function is smooth, probabilities $\hat{\boldsymbol{y}}$ are computed from softmax/sigmoid function which are both smooth and they are from the definition equal to 1 or 0 only in the limit case, which in practice does not occur. Previously mentioned activation functions are also smooth, except for ReLU in 0, but this is solved by additional definition of derivative in 0 to be 0.

Because of the computational restrictions with large datasets, we cannot use all data during each training iteration so instead stochastic gradient

descent (SGD) where a single point is randomly sampled from training data or Mini-batch SGD where small subset is sampled, is used to compute the gradient.

Mini-batch SGD

Inputs: neural network$f(\boldsymbol{x};\boldsymbol{\theta})$, learning rate α, loss function $L(\hat{y}, y)$, dataset $(\boldsymbol{x}, y)$

Repeat

Sample a minibatch of size m from $(\boldsymbol{x}, y)$

$\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i^m L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), y^{(i)})$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\boldsymbol{g}$

There are some problems associated with SGD. First of them is selection of the learning rate. Very small value can lead to very slow convergence to optimal $\boldsymbol{\theta}^*$ and large value can lead to divergence.

There were many proposals how to fix this problem, one of them was introducing momentum term (SGD with momentum and SGD with Nestorov momentum). Simply put, successive iterations of the algorithm utilize information from previous iterations to accelerate learning if the gradient pointed into the same direction as in the previous iteration. On the other hand if in a successive iteration gradient pointed into the opposite direction, momentum slows down the descent.

Momentum term essentially represents the degree of weighting decrease and each update step is equal to exponential moving average of past gradients. Momentum largely prevents divergence, but there is still a problem selecting the right learning rate. Even if we choose the right one, every parameter is updated with the same learning rate, which is not always desirable as we can demand parameters which are not updated frequently to have larger learning rates and vice versa for parameters which are updated often.

Algorithms with adaptive learning rate aim to solve this issue. *Adagrad*, adaptive gradient algorithm [24], allows different learning rates for each parameter by scaling gradient value in each iterations. Problem with *Adagrad* is that scaling term is increasing in each iteration consequently decreasing learning rate and eventually stopping the learning process. Adadelta and RMSProb are very similar algorithms which instead of ever increasing scal-

ing term use its exponential average, which prevents inevitable end of the learning.

Another algorithm called *Adam* - adaptive momentum - [25] builds upon Adadelta/RMSProb but also allows for different momentum for each parameter.

Algorithm 3 ADAM

Input: neural network $f(\boldsymbol{x};\boldsymbol{\theta})$, learning rate α, loss function $L(\hat{y}, y)$, dataset $(\boldsymbol{x}, y)$

small constant ϵ, momentum β_1, momentum β_2

$\boldsymbol{s}, \boldsymbol{r}, t \leftarrow 0$

Repeat

Sample a minibatch of size m from $(\boldsymbol{x}, y)$

$\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i^m L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), y^{(i)})$

$t \leftarrow t + 1$

$\boldsymbol{s} \leftarrow \beta_1 \boldsymbol{s} + (1 - \beta_1)\boldsymbol{g}$

$\boldsymbol{r} \leftarrow \beta_2 \boldsymbol{r} + (1 - \beta_2)\boldsymbol{g}^2$

$\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\beta_1^t}$

$\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\beta_2^t}$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\alpha}{\sqrt{\hat{\boldsymbol{r}}+\epsilon}} \hat{\boldsymbol{s}}$

Momenta β_1 and β_2 are usually in a $0.9 - 0.99$ range, ϵ is used just to prevent division by 0. Value of the learning rate α is less relevant because of Adam's properties. Parameters $\boldsymbol{s}$ and $\boldsymbol{r}$ can be understood as a biased estimate of the first and the second moment of the gradient which, particularly during earlier stages of training, tend to be biased towards 0. This is why they are scaled and $\hat{\boldsymbol{s}}$, $\hat{\boldsymbol{r}}$ are approximately unbiased estimates. For unbiasedness it is required that both first and second moments of the gradient are stationary, which is almost always incorrect. Nevertheless the resulting effect on the training, is that learning rate is lower in the beginning when the estimate of moments is not precise and gets gradually higher with the increase of precision. Following Adam, there have been various modifications to this algorithm, but it is still widely used for general problems.

3.2.3 Regularization

L1 and L2 regularization is used predominantly in regressions, while for neural networks other more effective methods are used.

Early Stopping

Because improving accuracy is often the only goal during NN's training, it tends to overfit and reach very high accuracy on the training set. This can mean that after the network finds out the right model it continues to learn on the specificities of the training set. This problem can often be solved by early stopping of the training. [26]

Dropout

Dropout is a simple yet effective regularization method introduced in Srivastava et al. in 2014 [27] . When applied to a layer of NN, each neuron is independently "dropped", that is, set to zero with probability p called dropout rate. This in effect pushes NN to try learning without using portion of both input and NN engineered features, which forces NN to better generalize.

Dropout is applied only during training and during testing all neurons are used. However, because at training time average number of used neurons in a given forward propagation is lower, they tend to have larger weights to compensate. That's why during testing, weights are scaled down by the factor *1-p* .

Batch normalization

During training, distribution of inputs from one layer to another tends to constantly change, because of the weights change. This slows down learning, because learning rate needs to be lowered in order to achieve better results. To counter this issue, Ioffe and Szegedy [28] proposed method in which for every batch, output of the layer is normalized by mean and variance of a output values of given batch, before it is passed to the next layer. This leads

into faster convergence, because we can increase learning rate, as we know that possible values in the activation functions won't be large (which can also solve the problem of the saturating non-linearities). At the same time process of normalization introduces noise into the layers, which helps with generalization.

During evaluation, data are normalized by mean and variance calculated by exponential moving average of batches used during training.

3.2.4 Model selection and hyperparameter tuning

Deep neural networks have capability to model complex relationships between inputs and as such, can perform feature selection themselves, however, because number of network parameters grows exponentially with number of features, at least some level of external feature selection is needed to achieve reasonable training time.

Hyperparameters in NN differ from NN parameters in that they are not being being trained during the training. In our case they include number of layers, number of neurons in each layer, learning rate, batch size etc. Most straightforward way to find optimal hyperparameters is **grid search**, which is just trying every combination of hyperparameters whose possible values are predefined by the user and have to be discrete. This process guarantees to find optimal set of parameters (if they are present in the predefined values), but is computationally infeasible.

Similar, but more effective approach is **random search** which samples vectors from predefined hyperparameter space (which can be continues). This approach is much faster and often as good at using grid search [29].

Previously mentioned techniques do not adjust as a response to the training results, as every iteration is independent. Advanced technique called **Bayesian optimization** uses past results to construct Gaussian process priors (with specific assumptions like similar vectors should yield similar results) to predict which regions of the hyperparameter space will have better results. At each iteration, Bayesian optimization faces exploration vs.

exploitation problem, that is, if the next sampled vector should be similar to previous good performing vector or if it should sample from areas of which there is none or little information about resulting performance. Theory behind Bayesian optimization is over the scope of this thesis and we encourage readers to read [30] [31] to get deeper understanding of this method.

3.3 Decision Trees

Decision trees (DT) [32] [33] are simple yet effective method for prediction. They split the predictor space into a number of non-overlapping smaller regions where every point in the given region has the same predicted value.

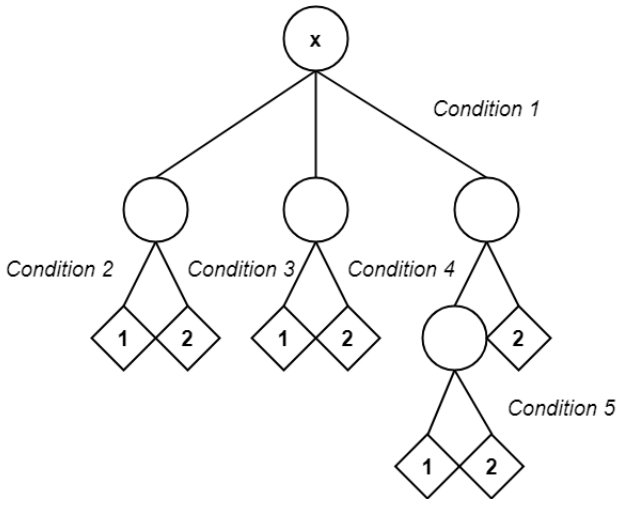

Figure 2: Simple decision tree
Inputs are classified into 2 classes based on the best selected conditions

Figure 2 shows simple decision tree. Input data $\mathbf{x}$ are split by Condition 1 (example of a simple condition can be a relational operator $\leq, \geq, =$ etc.) into 3 subsets, often called branches, which are further split each into 2 branches by Condition 2-4. Square nodes represent final classification nodes called terminal nodes or often leaves - all data points in the same terminal node are predicted as the same class. There are also instances when the tree decides to continue splitting some subset of data as can be seen on Condition 5.

Process of construction of optimal DT is a NP-complete problem, that is quickly computationally infeasible with the increasing number of features.

Instead of finding the most optimal DT, construction methods use *greedy* algorithms, which choose decision at each node based on immediate gain in performance. Using these algorithms does not guarantee convergence to the optimal DT but by using multiple different trees in an ensemble, this issue can be diminished.

CART (Classification and Regression Trees) algorithm uses consequent binary splits which lead into the highest loss decrease. In most implementations of CART, either Information gain characterized by entropy or Gini index of impurity is used as a loss function, which are both very similar measures.

During the tree construction, set of samples in a branch is split into 2 subsets, each having its own value of the Gini index, in a way to achieve the lowest value of the total Gini index. Gini index of impurity, in the binary classification problem, represents probability that randomly chosen datapoint in the given subset is misclassified if it is given probability of belonging to the positive (1) class equal to the proportion of this class in the subset.

$$Gini\ index = 1 - \sum_i p_i^2 = 1 - p^2 - (1-p)^2 = 2p(1-p), \tag{16}$$

where p_i is a proportion of class i in the subset and we use the fact that in the binary case if $p_1 = p \rightarrow p_2 = (1-p)$.

Therefore if we achieve perfect split, where all samples are separated then proportion of either class in one subset will be either 1 or 0 leading into the Gini index of 0. On the other hand if we split in a way that each subset has 50% of each class, then randomly chosen datapoint has a probability of 0.5 to be misclassified as it is given probability of belonging to the either class of 0.5 .

Total Gini index for binary splitting is the weighted sum of Gini indices for each region, weights correspond to proportion of total number of samples in the given region.

$$Total\ Gini\ index = \frac{n_1}{n}(2p_{11}p_{12}) + \frac{n_2}{n}(2p_{21}p_{22}), \tag{17}$$

where n_i is the number of datapoints in region i and p_{ij} is a proportion of classes j in a region i.

Split with the lowest Gini index is chosen until the all datapoints are separated, which is not optimal as it can very easily lead into overfitting.

To regularize the tree we can perform early stopping when we limit depth of the tree, we can limit the minimum number of samples in one terminal node or we can use tree pruning, which deletes last few layers of the tree. These methods can also act as a feature selection tool. For example, if we limit the depth, feature which is not relevant will not be split on.

Probably the best regularization technique is not to use a single decision tree, but instead use many trees in ensemble.

3.3.1 Ensembles of trees

Ensemble learning involves using multiple models, which decide together on the final prediction. Generally, ensembles outperform any single classifier in prediction accuracy [3]. Important requirement in order for ensembles to perform better than any individual model is to not have very high correlation in predictions between models. In this way mistakes of an individual model are fixed by the other models.

3.3.2 Bagging

Bootstrap aggregating commonly called bagging, was introduced by Breinman (1996) [34]. The whole process involves construction of multiple trees. For each one, training dataset is randomly sampled with replacement. Final probability is calculated as an average of probabilities of all trees. Bagging procedure can be used with any predictor, but it particularly improves decision trees models, as they are very data dependent and relatively small changes in data can result in widely different trees. Bagging addresses this problem by significantly reducing variance of the final predictor.

3.3.3 Random Forests

Random forests [35] provide improvement compared to bagged decision trees by decreasing correlation between models in bagged predictor.

Main idea behind RF is following [33]: In the case of bagged DT, expected value of the average of all predictions is equal to the expected value of any single tree, however the point of bagging is to decrease variance. In the ideal case, when predictions by each tree are i.i.d. with variance σ^2, their average has variance $\frac{1}{B}\sigma^2$, B being the number of trees in the ensemble. If they are not independent but still i.d. with pairwise correlation ρ, then their average has variance $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$. To reduce pairwise correlation of predictions, RF restricts number of possible features at each split.

Normally during bagging, DT can choose any feature at any split. In random forests set of all features with magnitude p is randomly reduced at each split, default size of this subset is $\sqrt{p}$. In comparison to bagging, this process ensures that if there are any strong predictors, they are not present in majority of trees and hence correlation between them is reduced.

3.3.4 Boosting

Additive modeling uses sum of simple functions to approximate an unknown complex function. Boosting follows this idea by using many weak learners (models with low predictive performance) to construct a well performing model (strong learner). In comparison to bagging tree models, construction of individual models used for adaptive boosting is not independent, because during the given model construction, information of past performance is used (weights on examples which were misclassified in previous models are increased and on correctly classified examples are decreased). By doing this process, later models are more focused on the mistakes made in the earlier models.

General formula for boosted model is

$$f(x) = \sum_{i=1}^{N} \beta_i f_i(x; \gamma_i) \tag{18}$$

where f_i are simple classification functions (weak learners) with parameters γ_i, and β_i are weights assigned to each function. In practice, shallow decision trees are most often used as f_i, as they are often sufficient weak learners. This is in essence the same formula as the one used to describe a layer of a neural network.

AdaBoost (versions M1,M2 and R)[36] was the first adaptive boosting algorithm, which could utilize any weak learner. This algorithm is also referred to as a Discrete Adaboost, as weak learners used to build the final classifier output classes instead of probabilities. For our purposes of comparison of predictive powers of different methods, we require algorithm called Real Adaboost [37] [33] which outputs probabilities.

Real Adaboost

Input: M weak learners $G_m(\boldsymbol{x})$, dataset $(\boldsymbol{x}, y)$ with N observations

Initialize observation weights $w_i = \frac{1}{N}, i = 1, 2, ..., N$

for $m = 1$ to M

Fit a classifier $G_m(\boldsymbol{x}) = \hat{P}_w(y = 1|\boldsymbol{x}) \in [0, 1]$ using weights w_i

Set $f_m(\boldsymbol{x}) = \frac{1}{2} \log \frac{G_m(\boldsymbol{x})}{1-G_m(\boldsymbol{x})}$

$w_i \leftarrow w_i \cdot e^{-y_i f_m(\boldsymbol{x}_i)}, i = 1, 2, ..., N$

Normalize w so that $\sum_i w_i = 1$

$G(\boldsymbol{x}) = sign(\sum_{m=1}^{M} f_m(\boldsymbol{x}))$

For this algorithm $y \in [-1, 1]$. Weight of weak learner $G_j(\boldsymbol{x})$ on the final prediction is expressed by f_j. Sign of f_j indicates class and magnitude indicates confidence in the prediction.

Gradient boosting introduces gradient descent similar to the one used in the neural networks. Similarly, it requires differentiable loss function to ensure existence of its gradient. Standard loss function used in the gradient boosted models for binary classification, is, as in the case of neural networks, binary cross entropy loss.

We let our simple functions output log odds instead of probabilities $\hat{y}$. This way, we can sum them to create an additive model. At the end we use sigmoid function to transform log odds to probabilities $\hat{y}_i$. Then cross entropy loss becomes

$$L(y, f(\boldsymbol{x})) = -\sum_i \left[y_i \log \frac{1}{1+e^{-f(x_i)}} + (1-y_i) \log \frac{e^{-f(x_i)}}{1+e^{-f(x_i)}} \right] \tag{19}$$

which can be transformed into

$$L(y, f(\boldsymbol{x})) = \sum_i f(x_i) y_i - \log\left(1 + e^{f(x_i)}\right) \tag{20}$$

with derivative

$$\frac{\partial L(y, f(x_i))}{\partial f(x_i)} = y_i - \frac{1}{1+e^{-f(x_i)}} = y_i - \hat{y}_i. \tag{21}$$

Gradient boosted trees for binary classification

Input: M weak learners $G_m(\boldsymbol{x})$, dataset $(\boldsymbol{x}, y)$ with N observations

Initialize $f_0(\boldsymbol{x}) = 0$

for $m = 1$ to M

Calculate pseudo-response $r_i = -\frac{\partial L(y, f_{m-1}(\boldsymbol{x_i}))}{\partial f_{m-1}(\boldsymbol{x_i})} = y_i - \frac{1}{1+e^{-f_{m-1}(\boldsymbol{x_i})}}, i = 1, 2, ..., N$

Fit a regression tree with targets r_i and J terminal nodes which outputs terminal regions $R_{jm}, j = 1, 2, ..., J$

$\gamma_{jm} = \text{argmin}_\gamma \sum_{\boldsymbol{x}_i \in R_{jm}} L(y_i, f_{m-1}(\boldsymbol{x_i}) + \gamma), j = 1, 2, ..., J$

$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \nu \gamma_{jm} I(\boldsymbol{x} \in R_{jm}), j = 1, 2, ..., J$

Output $f_M(\boldsymbol{x})$

Parameter ν is called shrinking parameter and is used for regularization. It was shown by Friedmann [38], that for small values $\nu \leq 0.1$, final models are more generalized.

Small modification to this algorithm called *Stochastic gradient boost*, came as a reaction to the performance of the models (particularly bagging) which introduce randomness into their construction process [39]. This procedure was shown to be especially effective in case of noisy datasets. Stochastic gradient is very similar to the Mini-batch SGD in the neural networks. During one iteration of the algorithm, only a random subset of size $\tilde{N} \leq N$ of datapoints is used to update the final function. Friedmann's research indicates that on average, stochastic gradient boost improves performance over the basic gradient boost.

3.4 Evaluation

For the small German Credit dataset, standard k-fold cross-validation is used to score the models [40]. Whole dataset is shuffled and split into k even groups, then k models are constructed using $k-1$ groups as training set and the last group as test set. For the reason of imbalanced dataset, we use stratified k-fold method which ensures that testing set has similar class distribution as the original dataset.

In the case of LC, training and testing split are used (70%:30%), which means models are trained on the training set and hyperparameters are tuned according to performance on testing split. Only after selecting the best models, k-fold cross validation is used created again by the stratified k-fold method.

Metrics

Accuracy is the most widely used metric to represent predictive abilities of the model, but it can be often deceptive. In the case of highly imbalanced classes, high accuracy can be reached by simple model always predicting the largest class. Accuracy itself assumes we value absolute number of true positives the same way as true negatives. This is however not always the case as it isn't in the case of loans. We argue that in this case precision, recall and specificity are better metrics compared to the accuracy alone.

Precision (also called positive predictive value) is the ratio of correctly

positively predicted cases and all positively predicted cases, while **recall** (also called true positive rate TPR or sensitivity) represents ratio of correctly positively predicted cases and all positive cases. **Specificity** (also called true negative rate) measures fraction of correctly predicted negative cases.

$$precision = \frac{TP}{TP + FP} \quad recall = \frac{TP}{TP + FN}$$

$$specificity = \frac{TN}{TN + FP}$$

Receiver operating characteristic (ROC curve) represents points of all possible values of true positive rate (recall) and false positive rate (1-specificity). **ROC area under curve** (ROC AUC) captures the whole range of possible decision thresholds into one score. AUC represents probability, that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance [41]. AUC of 0.5 represents "random guessing" model and AUC of 1 represents perfect model with 100% true positive rate and 0% false positive rate.

Threshold

All used classification models output probabilities instead of classes and threshold value of prediction has to be chosen. Most models, on default, choose threshold 0.5 and predict cases with $p > 0.5$ as positive and $p < 0.5$ as negative. By decreasing this threshold, we raise the recall but at the same time decrease the precision and specificity. In the lending environment, where declining good lender costs very little compared to the accepting bad lender, recall is more important than precision. At the same time it is crucial to balance recall and specificity. It is on the every lending institution's own discretion to balance these factors.

One possibility to find better threshold, without introducing our subjective judgment or without having specific financial information about the lending institution, is through maximization of metric called Informedness

[42], also called Youden's J statistic for binary classification.

$$J = TPR + TNR - 1 \tag{22}$$

On the *Figure 3* we can see ROC curve and maximized J statistic. Vertical line represents longest distance between "random guess" and ROC curve, their intersection represents maximized J. Informendness is the probability that a prediction is informed and not just a random guess, where value of 1 is perfect classification and 0 random chance, but unlike ROC, J is a function of the threshold.

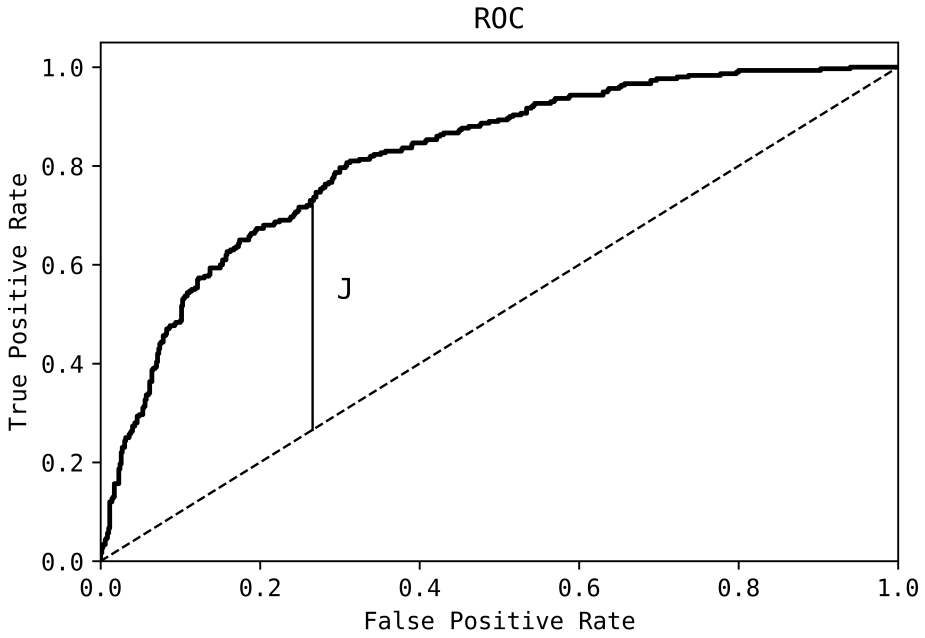

Figure 3: ROC curve and Informendness J
Intersection of the longest line between the "random guess" line and ROC curve maximizes value of the J.

Our objective is to find models with highest ROC AUC, which is threshold independent, then to better illustrate the results we find threshold by maximizing Informedness J.

3.4.1 Test for comparing models

Due to our objective metric ROC AUC and not the accuracy, we cannot use statistical tests, which are based on confusion matrix (for example McNe-

mar's test).

Natural choice would be to use k-fold cross-validation on the same folds for both models to get $2k$ metric values and then, use the paired two-sample t-test. However t-test requires independence of observations in each sample, which is not true, as during cross-validation, each datapoint is used $k-1$ times to calculate final sample of metrics.

Dietterich (1998) [43] showed that using this process to perform t-test leads into high Type I error, that is, excessive rejecting of the null hypothesis.

To compare 2 models, Dietterich suggests using method called 5x2-fold cross-validation, which offers compromise between low Type I and II errors and relative speed of evaluation. In this method, 5 different pairs of folds are used to train 2 models (20 models in total). The requirement for independence between samples in each 2-fold cross-validation repetition is still violated, however, experimental results show, that it doesn't cause large increase of errors.

Nadeau and Bengio (2003) [44] argue that this method underestimates variance and they propose improvement by proposing correction of t-statistic, which leads into low Type I and II errors and at the same time allows to use $g$xk-fold validation with $k > 2$.

If we let $x_{ij} = m_{ij}^1 - m_{ij}^2,\ i = 1, 2, ...g,\ j = 1, 2...k$ represent set of differences from metrics calculated by model 1 and 2 for the same fold j in the g-th k-fold cross validation, then corrected t-statistic which follows t-distribution with $gk-1$ degrees of freedom is

$$t = \frac{\hat{\mu}}{\sqrt{(\frac{1}{gk} + \frac{n_2}{n_1})\hat{\sigma}^2}}, \tag{23}$$

where $\hat{\mu}$ and $\hat{\sigma}^2$ are the estimates of mean and variance of the set of x_{ij} and $\frac{n_2}{n_1}$ is a ratio of the number of samples used in the testing and the number of samples used in the training. Null hypothesis used for this test is $H_0 : \mu = 0$

.

In our experiments, we will use corrected t-test with 20x4-fold cross-validation in case of the German dataset and 5x2-fold cross-validation in case of LC dataset due to its size.

4 Datasets

First, we use German Credit dataset [45], which is widely used for benchmarking binary classification models. It contains 1000 observations with 20 features (24 including dummy variables) classifying people as good (70%) or bad (30%) credit risks.

Second dataset is provided by Lending Club, P2P US lending platform [46]. LC acts as an intermediary for personal loans up to $40,000. It rates loan applicants by 7 ratings A-G, with corresponding average interest rate $\approx 7 - 25\%$. Potential investors can select level of risk they are willing to face and LC builds diversified portfolio of loans for them. Another way is to directly choose loan from the marketplace where all unmatched or not fully financed loans are placed. Although the business model of P2P platform is not identical to the one of a bank, our goal is simply to compare different classification models. Additionally Lending Club (LC) is one of the few institutions that provide large public data.

This dataset contains data on loans with completed status from January 2015 to December 2017, which is 448,890 loans of which 76% loans are fully paid and 24% are in default, that is payments are 121+ days past due and repayment is very unlikely. Each loan has 116 features, however large amount of them is not usable because they are mostly empty, they were recorded after the loan was accepted or they are features engineered by LC's internal models (rating category and interest rate). Few of the potentially important features are location, length of employment, number of delinquencies, home ownership, current debt, credit score and other financial indicators.

LC offers two loan terms, 36 or 60 month but doesn't provide exact date of loan start, which means potential time frame of approval of loan application with fully paid status is January 2010 - December 2014. Simultaneously, because of its unpredictable nature, potential time frame for loan application with the default status can be anywhere between the middle of 2009 to the middle of 2017. It is however reasonable to assume, that occurrence of defaults is more prevalent in the early to mid stages of loan terms rather

than very close to the end.

The reason why this time frame is important, are underlying economic conditions. This is because large external shock can be a cause of default even for people who would normally have no payment problems. Assuming majority of applications came in between January 2010 and December 2016, economy of the USA was recovering from 2008-09 financial crisis with continuous decline of unemployment from its high in 2010 of 9.1% to the 4.9% in 2016 [47], GDP growth oscillated around 2% with highs of 2.7% in 2010, 2013 and 2014 and lows of 1.3% in 2012 [48]. Overall, economic conditions were potentially helpful for borrowers and there can be a case when if the exact same person applied in 2010 his propensity to default would be higher than if he applied in the end of 2015. However, this problem cannot be easily solved, as we do not know dates of loan applications, and we disregard it in our models.

Another issue is LC's selection process and its own credit risk models. It is reasonable to assume that LC tries to eliminate all easily predictable defaults by continuously improving its own models, so that ceteris peribus occurrence of defaults should be decreasing with time. This is again serious problem, which skews the data, as we train only on the subset of the population, which was accepted by LC, therefore results can't be extended to the whole population, which nevertheless, is not our goal.

5 Results

5.1 German dataset

In the case of logistic regression model, recursive feature elimination together with L2 regularization and just plain L1 regularization were used to select optimal feature set. Both models selected 21 from 24 features. For the tree-based methods manual feature selection didn't significantly improve the models, as these models have built-in regularization during model construction, which also acts as a feature selection. In neural network models, manual feature selection did not improve the performance, as the training itself should select only the relevant features.

	ROC AUC	Acc. %	Recall %	Specificity %	Prec. %	J
Logistic	0.7951 (0.0243)	72.22 (2.66)	71.72 (2.68)	72.44 (2.70)	52.81 (3.34)	0.4415 (0.0530)
NN	0.7943 (0.0238)	72.04 (2.67)	71.27 (2.67)	72.37 (2.70)	52.60 (3.33)	0.4364 (0.0531)
DT	0.7313 (0.0279)	70.52 (2.62)	58.40 (6.32)	75.71 (3.34)	50.84 (3.87)	0.3411 (0.0637)
Bagged DTs	0.7958 (0.0206)	72.82 (2.24)	71.97 (2.27)	73.19 (2.29)	53.56 (2.90)	0.4515 (0.0446)
RF	0.7938 (0.0216)	72.60 (2.56)	71.37 (2.65)	73.13 (2.58)	53.32 (3.26)	0.4450 (0.0514)
Adaboost	0.7875 (0.0223)	72.27 (2.62)	69.90 (2.75)	73.29 (2.71)	52.96 (3.42)	0.4319 (0.0521)
GB Trees	0.7903 (0.0207)	71.91 (2.14)	71.18 (2.14)	72.22 (2.20)	52.40 (2.69)	0.4340 (0.0424)
Ensemble	0.8065 (0.0215)	73.40 (2.71)	72.83 (2.61)	73.64 (2.83)	54.32 (3.51)	0.4647 (0.0532)

Table 1: Performance of the best models on German dataset

Even thought we tested deep neural network architectures with up to 5 layers, best networks had only 1 or 2 layers with 40-300 neurons and mostly used hyperbolic tangent as an activation function, which suggests there are no strong non-linearities present. For regularization, best models used batch normalization and dropout with a rate 0.3-0.5 .

Single decision trees model were regularized by limiting minimal number of samples in terminal nodes, while bagged decision trees and gradient boosted trees were limited in number of samples used to construct single decision tree. No regularization was needed during random forests construction. Interestingly, the best Adaboost ensemble was trained using random forests made out of shallow decision trees instead of just using single decision trees.

Table 1 presents results of the best models and their standard deviations. All single models, with the exception of a single decision tree model, performed similarly and their results were not statistically different from each other. The single decision tree model was, as expected significantly worse than other models.

The best model, which was better than all other single models at 5% significance level, was created by ensembling all the best models from each method except for the single DT. Final probability of default was calculated as an average of probabilities from each model, this method is also called voting or stacked ensemble. Although this difference was significant, it lead to only $\approx 1\%$ larger value of ROC AUC than other models, which is relatively minor improvement.

Looking at other metrics of the stacked ensemble model, which were chosen to maximize value of Informendness J, on average 72.83% of defaults was successfully predicted, 73.64% of non-default cases were predicted, however only 54.32% of samples which were predicted as defaults were really defaults, which means that almost half of potential good borrowers would be rejected.

5.2 LC dataset

Similarly to the German dataset, we performed feature selection for the majority of models, however, it did not improve performance.

The best neural network models had 2-5 layers, between 300 and 1500 neurons in each layer and used mostly ReLU activation function. Additionally, networks with batch normalization and dropout rate of 0.3-0.6 performed better that those without them.

The best gradient boosted trees model was heavily regularized by limiting depth, maximum number of features considered at each split and amount of training data used in the construction of a single weak learner decision tree.

As was the case in previous results, the best Adaboost ensemble model was constructed by random forests with shallow decision trees.

	ROC AUC	Acc. %	Recall %	Specificity %	Prec. %	J
Logistic	0.7172 (0.0011)	65.71 (0.12)	65.71 (0.12)	65.71 (0.12)	37.87 (0.12)	0.3142 (0.0023)
NN	0.7290 (0.0007)	66.62 (0.07)	66.62 (0.07)	66.62 (0.07)	38.83 (0.07)	0.3323 (0.0013)
DT	0.6823 (0.0011)	63.27 (0.14)	62.52 (0.25)	63.51 (0.17)	35.28 (0.14)	0.2603 (0.0031)
Bagged DTs	0.7115 (0.0009)	65.66 (0.18)	64.26 (0.32)	66.11 (0.32)	37.62 (0.12)	0.3037 (0.0012)
RF	0.7194 (0.0008)	65.96 (0.09)	65.51 (0.12)	66.10 (0.12)	38.07 (0.08)	0.3161 (0.0015)
Adaboost	0.7171 (0.0008)	66.07 (0.07)	65.03 (0.24)	66.40 (0.12)	38.10 (0.07)	0.3143 (0.0018)
GB Trees	0.7319 (0.0009)	66.83 (0.07)	66.83 (0.07)	66.83 (0.07)	39.05 (0.08)	0.3366 (0.0014)
Ensemble	0.7303 (0.0010)	66.71 (0.10)	66.71 (0.10)	66.71 (0.10)	38.93 (0.11)	0.3342 (0.0021)

Table 2: Performance of the best models on Lending Club dataset

Table 2 presents results of the best models and their standard deviations. Variance in the results was, compared to the German dataset, very low, which is primarily caused by the size of LC dataset.

Overall, best results were obtained by the gradient boosted trees model, which was better than all models, including stacked ensemble, at 1% significance level. The next best results were obtained by neural network, which was better than all models at 1% significance level, with the exception of stacked ensemble model, which was worse at the 5% significance level, and obviously GB trees model.

The third best model was the stacked ensemble model, which had higher ROC AUC then remaining models at 1% significance level.

The worst model was, again, a single decision tree. From the remaining tree-based models, random forests slightly outperformed other models. Our baseline logistic regression model outperformed only bagging and single decision tree at 1% significance level.

If we take closer look at the our best model performance after choosing threshold, we see, that it is worse than in the German dataset case. Overall, we were able to predict 66.83% of all defaults and 66.83% of all fully paid loans, but of all predicted defaults almost 60% were predicted wrongly. This may seem like a very large fraction, similar to the German dataset, but as we already mentioned, without knowing further loan details, we can't objectively evaluate this number.

6 Conclusion

This thesis tries to introduce the task of classifying potential borrowers as "good" and "bad" depending on the probability of their potential default in the case they are granted a loan. Overview of the credit scoring history together with introduction to the internal ranking models used in the lending institutions is presented in order to acquaint the reader about often used methods.

In order to construct models, two datasets are selected. German dataset which is the most often used dataset in credit modeling literature and dataset from the Lending Club, P2P lending platform, which to our knowledge offers the largest current publicly available dataset related to credit scoring.

Models explored are logistic regression as a baseline model, which is often used for credit scoring, neural networks, which were in the last few years greatly improved by the new advances in the deep learning field and tree-based model ensembles, which utilize simple decision trees to construct more complex models.

Contrary to the most studies involving this task, average accuracy is not used as the primary indicator of the model quality. Instead, ROC AUC statistic is used, which in essence captures the whole range of possible accuracies which could be achieved by the given predicted probabilities.

Overall results on the smaller German dataset suggest that no single model has significantly better predictive performance and they are all equally capable. However after the all single models are combined into a stacked ensemble model, results are significantly better than any single model.

Results on the larger LC dataset are again fairly similar, with the best model gradient boosted trees, differing only by $\approx 2\%$ with the worst model. Neural networks achieve above average performance, while stacked ensemble performs worse than the best single model.

Even if there are cases when stacked ensembles or neural networks beat any single model, they achieve $\approx 1\%$ more ROC AUC score than logistic regression. Given the time requirements for training and tuning the networks,

there could plausibly exist cases when logistic regression is preferred to the neural networks or stacked ensembles even if it is worse.

Further research should focus on directly tying loss function used in the algorithms with potential losses or gains in order to find out the method to find the most profitable decision threshold. Underlying economic factors together with the selection bias problem, should also be incorporated into the scoring process. Research focused only on the models created by ensembling could explore how to most optimally construct single models whose sole purpose is to act as a part of a stacked ensemble model and they could also explore application of bagging and boosting algorithms on neural networks.

References

[1] Jodi L Bellovary, Don E Giacomino and Michael D Akers. "A review of bankruptcy prediction studies: 1930 to present". In: *Journal of Financial education* (2007), pp. 1–42.

[2] Francisco Louzada, Anderson Ara and Guilherme B Fernandes. "Classification methods applied to credit scoring: Systematic review and overall comparison". In: *Surveys in Operations Research and Management Science* 21.2 (2016), pp. 117–134.

[3] Bhekisipho Twala. "Multiple classifier application to credit risk assessment". In: *Expert Systems with Applications* 37.4 (Apr. 2010), pp. 3326–3336. ISSN: 09574174. DOI: 10.1016/j.eswa.2009.10.018.

[4] Huawei Chen et al. "Ensembling extreme learning machines". In: *International Symposium on Neural Networks*. Springer. 2007, pp. 1069–1076.

[5] Christopher Krauss, Xuan Anh Do and Nicolas Huck. "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500". In: *European Journal of Operational Research* 259.2 (2017), pp. 689–702.

[6] David J Hand and William E Henley. "Statistical classification methods in consumer credit scoring: a review". In: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 160.3 (1997), pp. 523–541.

[7] David West. "Neural network credit scoring models". In: *Computers & Operations Research* 27.11-12 (2000), pp. 1131–1152.

[8] Bart Baesens et al. "Using neural network rule extraction and decision tables for credit-risk evaluation". In: *Management science* 49.3 (2003), pp. 312–329.

[9] Maher Alaraj, Maysam Abbod and Ziad Hunaiti. "Evaluating Consumer Loans Using Neural Networks Ensembles". In: *International Conference on Machine Learning, Electrical and Mechanical Engineering*. 2014.

[10] Belás Jaroslav and Cipovová Eva. "Internal Model of Commercial Bank as an Instrument for Measuring Credit Risk of the Borrower in Relation to Financial Performance (Credit Scoring and Bankruptcy Models)". In: (), p. 17.

[11] ECB Banking Supervision. "Guide for the Targeted Review of Internal Models (TRIM)". en. In: (Feb. 2017), p. 155. URL: https://www.bankingsupervision.europa.eu/ecb/pub/pdf/trim_guide.en.pdf.

[12] Oesterreichische Nationalbank. "Guidelines on credit risk management". In: *Rating models and validation, Wien* (2004).

[13] Mostafa Mahmoud, Najla Algadi and Ahmed Ali. “Expert system for banking credit decision”. In: *Computer Science and Information Technology, 2008. ICCSIT'08. International Conference on*. IEEE. 2008, pp. 813–819.

[14] Maja Pohar, Mateja Blas and Sandra Turk. “Comparison of logistic regression and linear discriminant analysis: a simulation study”. In: *Metodoloski zvezki* 1.1 (2004), p. 143.

[15] Jonathan N Crook, David B Edelman and Lyn C Thomas. “Recent developments in consumer credit risk assessment”. In: *European Journal of Operational Research* 183.3 (2007), pp. 1447–1465.

[16] Deloitte. “Credit scoring - Case study in data analytics”. en. In: (Apr. 2016), p. 18. URL: `https://www2.deloitte.com/content/dam/Deloitte/global/Documents/Financial-Services/gx-be-aers-fsi-credit-scoring.pdf`.

[17] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[18] Kurt Hornik, Maxwell Stinchcombe and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.

[19] Kurt Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural networks* 4.2 (1991), pp. 251–257.

[20] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[21] Andrew M Saxe, James L McClelland and Surya Ganguli. “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks”. In: *arXiv preprint arXiv:1312.6120* (2013).

[22] Xavier Glorot, Antoine Bordes and Yoshua Bengio. “Deep sparse rectifier neural networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011, pp. 315–323.

[23] Andrew L Maas, Awni Y Hannun and Andrew Y Ng. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. icml*. Vol. 30. 1. 2013, p. 3.

[24] John Duchi, Elad Hazan and Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization”. In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159.

[25] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[26] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[27] Nitish Srivastava et al. "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[28] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[29] James Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization". In: *Journal of Machine Learning Research* 13.Feb (2012), pp. 281–305.

[30] Jonas Mockus. "Application of Bayesian approach to numerical methods of global and stochastic optimization". In: *Journal of Global Optimization* 4.4 (1994), pp. 347–365.

[31] Eric Brochu, Vlad M Cora and Nando De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning". In: *arXiv preprint arXiv:1012.2599* (2010).

[32] L Breiman et al. "Classification and regression trees". In: (1984).

[33] Hastie Trevor, Tibshirani Robert and Friedman JH. *The elements of statistical learning: data mining, inference, and prediction*. 2009.

[34] Leo Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[35] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[36] Yoav Freund and Robert E Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of computer and system sciences* 55.1 (1997), pp. 119–139.

[37] Jerome Friedman, Trevor Hastie, Robert Tibshirani et al. "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)". In: *The annals of statistics* 28.2 (2000), pp. 337–407.

[38] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[39] Jerome H Friedman. "Stochastic gradient boosting". In: *Computational Statistics & Data Analysis* 38.4 (2002), pp. 367–378.

[40] Hal R Varian. "Big data: New tricks for econometrics". In: *Journal of Economic Perspectives* 28.2 (2014), pp. 3–28.

[41] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[42] David Martin Powers. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation". In: (2011).

[43] Thomas G Dietterich. "Approximate statistical tests for comparing supervised classification learning algorithms". In: *Neural computation* 10.7 (1998), pp. 1895–1923.

[44] Claude Nadeau and Yoshua Bengio. "Inference for the Generalization Error". In: *Machine Learning* 52.3 (2003), p. 239.

[45] Professor Dr. Hans Hofmann. *Statlog (German Credit Data) Data Set*. 1994. URL: `https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)`.

[46] *Lending Club Statistics*. URL: `https://www.lendingclub.com/info/download-data.action`.

[47] U.S. Bureau of Economic Analysis. *Civilian Unemployment Rate [UNRATE]*. 2018. URL: `https://fred.stlouisfed.org/series/UNRATE`.

[48] U.S. Bureau of Economic Analysis. *Percent Change of Gross Domestic Product [CPGDPAI]*. 2018. URL: `https://fred.stlouisfed.org/series/CPGDPAI`.