**Vysoká škola ekonomická v Praze**

Fakulta financí a účetnictví

katedra bankovnictví a pojišťovnictví

Finanční inženýrství

**DIPLOMOVá práce**

2024 Stevan Vujčić

**Vysoká škola ekonomická v Praze**

Fakulta financí a účetnictví

katedra bankovnictví a pojišťovnictví

Finanční inženýrství

**Komparativní analýza modelů pro predikci pravděpodobnosti defaultu**

Autor diplomové práce: Stevan Vujčić

Vedoucí diplomové práce: Milan Fičura

Rok obhajoby: 2024

Čestné prohlášení

Prohlašuji, že jsem diplomovou práci na téma „Komparativní analýza modelů pro predikci pravděpodobnosti defaultu“ vypracoval samostatně a veškerou použitou literaturu a další prameny jsem řádně označil a uvedl v přiloženém seznamu.

V Praze dne doplnit datum

.................................................................................

Stevan Vujčić (výše podepsat)

Abstrakt

(stručný obsah práce s uvedením hlavních výsledků v maximálním rozsahu 15 řádků)

Klíčová slova

Abstract

Abstract in English.

Key words

Key words in English

Table of Contents

[Introduction 6](#_Toc162644791)

[1 Theoretical Framework 7](#_Toc162644792)

[1.1 Institutional Environment 7](#_Toc162644793)

[1.1.1 EBA Note on the Use of Machine Learning for IRB models 7](#_Toc162644794)

[1.2 Literature Review 8](#_Toc162644795)

[2 Overview of Modeling Methodology 9](#_Toc162644796)

[2.1 Model Estimation Techniques 9](#_Toc162644797)

[2.1.1 Logistic regression 9](#_Toc162644798)

[2.1.2 Neural Network 11](#_Toc162644799)

[2.1.3 Support Vector Machines 15](#_Toc162644800)

[2.1.4 Tree-based Methods 16](#_Toc162644801)

[2.1.5 KNN 19](#_Toc162644802)

[2.2 Model Testing 20](#_Toc162644803)

[2.2.1 Accuracy Ratio 20](#_Toc162644804)

[2.2.2 Area Under the Curve 21](#_Toc162644805)

[2.3 Data Preprocessing 21](#_Toc162644806)

[2.3.1 Oversampling 21](#_Toc162644807)

[2.3.2 Data Transformations 22](#_Toc162644808)

[2.3.3 Forward feature selection 22](#_Toc162644809)

[3 Application on Czech Mortgages Portfolio 23](#_Toc162644810)

[3.1 Modeling 23](#_Toc162644811)

[3.1.1 Preprocessing: data cleaning 24](#_Toc162644812)

[3.1.2 EDA 24](#_Toc162644813)

[3.1.3 Preprocessing: transformations and feature selection 25](#_Toc162644814)

[3.1.4 Logistic regression 29](#_Toc162644815)

[3.1.5 Neural Network 30](#_Toc162644816)

[3.2 Comparison of Results 33](#_Toc162644817)

[Final Remarks 34](#_Toc162644818)

[References 35](#_Toc162644819)

[Graphics and Tables 36](#_Toc162644820)

[Appendices 37](#_Toc162644821)

# Introduction

In this thesis, a comparison of 2 probability of default (PD) estimation methods is outlined. The traditional logistic regression approach is contrasted against Artificial Neural Networks (ANN).

Write about: credit risk environment, conceptual advantages vs disadvantages, datasets, results, more transparent modeling procedure than in articles.

Abbreviations list:

EBA – European Banking Authority

ML – Machine Learning

IRB – Internal Ratings-based

GDPR – General Data Protection Regulation

AI – Artificial Intelligence

PD – Probability of Default

LGD – Loss Given Default

EaD – Exposure at Default

LTV – Loan-to-Value

# Theoretical Framework

In the following subsections, a systematical approach to finding the relevant literature published so far is outlined. Then, relevant findings are summarized.

## Institutional Environment

Text.

### EBA Note on the Use of Machine Learning for IRB models

In August 2023, EBA published EBA/REP/2923/28 – “Follow-up report from the consultation on the discussion paper on machine learning for IRB models”. The document addresses the question of how ML techniques fit into the IRB framework. It also takes a broader look from the standpoint of GDPR regulation and the AI act. As from the context of the document, ML techniques here refer to non-traditional modeling approaches (e.g. artificial neural networks). The main findings/conclusions of the report are summarized in the next paragraphs.

EBA reports that the use of ML models is still at a developmental stage in the industry. When used, ML techniques are employed by financial institutions for PD modeling. Conversely, they are not used or are used to a lesser extent when other parameters such as EaD or LGD are being modeled. Furthermore, it is reported that financial institutions use ML in the pre-model estimation phase of development. For example, inputting missing data or feature selection. Also ML techniques were reported to be used as benchmarks for traditional models. Financial institutions do not necessarily refrain from the use of traditional approaches. However, once ML is employed to set the performance bar, this feedback is useful input into making a conclusion about whether the traditional model itself should be better.

A concern is raised about the interpretability of these estimation techniques as compared to a traditional model such as the logistic regression. It often cannot be with straightforwardness and ease concluded how the individual model features contributed to the model from the standpoint of economic interpretation.

It is claimed that financial institutions are struggling to deal with the tendency of ML models to overfit the data. Therefore, financial institutions should, according to EBA, invest special attention to in-sample and out-of-sample comparisons when using ML.

Financial institutions are reported to not always have mature data infrastructures and procedures that handle the complexity that comes with the decision to use these techniques. This especially plays a role in the case of older data which was not collected not stored in such detail.

The EBA comments that financial institutions should understand the fact that the determination of capital requirements is being harmonized across the continent. To interpret this point, this means that the even larger diversity of modeling solutions that ML would inherently bring at least conceptually goes against this effort.

Finally, the EBA concludes that the financial institutions globally agree with the recommendations of the underlying discussion paper. Consequently, the EBA sets that the discussion paper should be used as the basis for ML modeling in risk applications.

## Literature Review

The scope of the relevant literature collection is defined as any literature that compares different estimation techniques in the field of probability of default modeling. This is aimed at finding relevant (1) research papers published in academic journals, (2) publications from financial institutions as well as (3) other insightful sources that are presented in other formats.

Table 1. Literature search results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Search engine | Criteria | # of results | # of referenced results | Query link |
| Web of Science | 1. keywords: machine learning credit scoring; 2. filter: “4.61.1820 Credit Scoring” 3. filter: from November 2013 to November 2023 | 315 |  | https://www.webofscience.com/wos/woscc/summary/adb5bbe9-768c-4a8d-967c-332fa6b5e22b-b17545fd/sort-group-background-citingcount/1 |
| Scopus | probability of default alternative methods | 132 |  |  |

# Overview of Modeling Methodology

In this section, the basics of estimation techniques and model testing used for default event classification problem and the performance tests are commented.

## Model Estimation Techniques

In this subsection, the estimation techniques which are used in the application section are reviewed.

### Logistic regression

The logistic regression is used as the industry standard due to its sound performance and good interpretability of its results. Consider the probability of default of the *i*-th observation *pi*, then the probability of default conditioned by right-hand side variables **x**i is:

|  |  |
| --- | --- |
| , | # |

where **β** is the vector of coefficients, *ui* is the error term for which a 0 mean and a distribution are being assumed. Since can take upon values from minus infinity to plus infinity, *Fi* is a link function that is used to confine that result between 0 and 1. In that way the result is converted into probability (Witzany, 2017). Witzany (2017) further argues that the deterministic part of the equation can be interpreted as the currently observable debtor’s credit capacity whilst the error term reflects the unknown development in future.

The choice of the link function can vary. The 2 basic ones are Probit and Logit. The 2 link functions give very similar results in econometric applications. Nevertheless, Witzany (2017) highlights that the logistic distribution has heavier tails, which can be seen as being more conservative since it assigns a larger weight to extreme events. The choice of the link function ultimately marks what assumption about the distribution of the error term is being made. In the case of Logit, the link function is the sigmoid curve:

|  |  |
| --- | --- |
| , | # |

where *z* equals the model specification introduced above: . The sigmoid curve is depicted below.

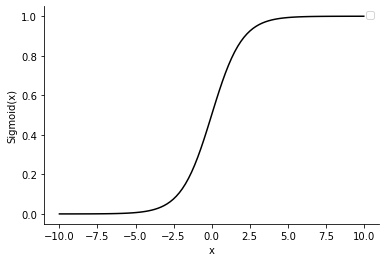


Figure 1: sigmoid function, source: author

To solve the equation defined above, the maximum likelihood estimator is used. In its log-transformed form:

|  |  |
| --- | --- |
| , | # |

where *L* is the likelihood function and **b** is the estimation of **β**. Since the first equation # holds, it also holds that the good-bad odds are:

|  |  |
| --- | --- |
| , | # |

where *pi* equates . Taking logarithms of both sides of the equation above yields the log good-bad odds:

|  |  |
| --- | --- |
| . | # |

Now, in the case that the sigmoid function is used, then the partial derivatives of the log-likelihood function are found. Here lecture notes of Piech (2017) are adjusted towards the notation used throughout this section and slightly more operational detail is provided. Starting from # above, the derivative of the likelihood function w.r.t. the parameter *bj* from **b** is:

|  |  |
| --- | --- |
| . | # |

First derive the logarithmic terms using the derivative of log rule and isolate the underived residuum:

|  |  |
| --- | --- |
| . | # |

Using the derivative of as the outer function and the derivative of as the inner:

|  |  |
| --- | --- |
| , | # |

where *xi,j* is a scalar. The above can be manipulated by:

|  |  |
| --- | --- |
| , | # |

further into,

|  |  |
| --- | --- |
| , | # |

finally, the denominator and the numerator below can partially cancel out,

|  |  |
| --- | --- |
| , | # |

and result in:

|  |  |
| --- | --- |
| . | # |

Witzany (2017) further states that given the negatively definite Hessian matrix the solution exists and it is unique. There are a variety of algorithms to solve the equation, one of them is the Newton-Raphson’s according to Witzany (2017).

### Neural Network

Consider a vector of input data **x** of *m* rows and 1 column. The vector is multiplied with a weights vector **w** that has 1 row and *m* columns (**wx**). The resulting linear combination is a scalar. The scalar result is then used as an input for the so-called activation function *f*. This function is chosen to be non-linear, e.g. sigmoid. The resulting calculation from the non-linear function is an activation – *a*.

The operations described in the previous paragraph create one neuron, which is the basic structure in this model. A neuron is understood as (1) a node characterized by an activation function that (2) uses linearly combined input from the previous layer to produce an output. It follows that one neuron is defined as follows:

|  |  |
| --- | --- |
|  | # |

where **w** is (*1,m*) and **x** is (*m,1*) and *f* is an arbitrary non-linear function. Schematically this is depicted in Figure X, where each line represents a multiplication and the scalars subscripted with *i* are entries of matrices **x** and **w** in the definition above. The rectangle represents the neuron *a*.

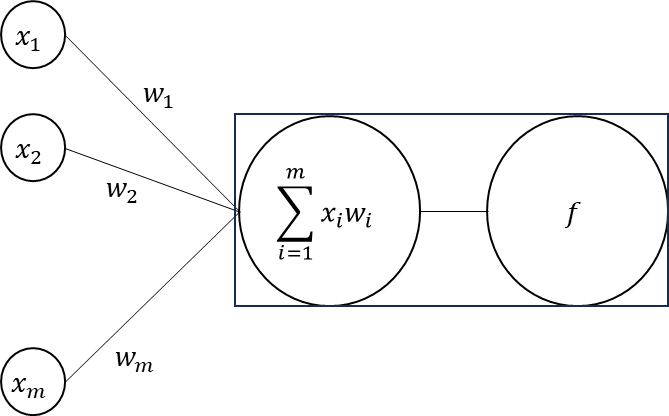


Figure 2: structure of a neuron, source: author, Witzany (2017)

Now, one neuron as defined above is part of a larger model structure. Consider that the neuron *a* defined above is just one of many – *ak* neurons. Together they form a vector of activations **a**. The vector **x**, being the vector of input data, remains unchanged. However, for every *ak* in the vector **a** to be treated separately, each *ak* is multiplied by a separate set of weights – **wk**. Hence, the vector **w** from the initial example is replaced by a matrix **W**. The dimensions of this matrix are (*k,m*) – the number of rows corresponding to the number of activations they are being fed towards and the number of columns corresponding to the number of rows of the vector **x**. It follows that this augmentation results in:

|  |  |
| --- | --- |
| , | # |

where the notation and dimensions are explained in the paragraph above. Extending the somewhat general form written above to multiple sets of activations, i.e. hidden layers boils down to simple stacking. For example, a neural network of 3 hidden layers would have:

|  |  |
| --- | --- |
| , | # |

where is **a1** – the first layer, or more clearly is **a2** – the second layer and so on. Finally, *y* is the output. The example developed above works only with a vector of **x**, i.e. one observation. For practical reasons it is more realistic to expect multiple observations:

|  |  |
| --- | --- |
| , | # |

where the change boils down to *y* becoming the vector **y**. Figure X shows a neural network of 3 input features, 3 activation nodes and one output. The matrix **W1** contains the weights connecting the input and the hidden layer whereas the vector **w2** leads to the output.

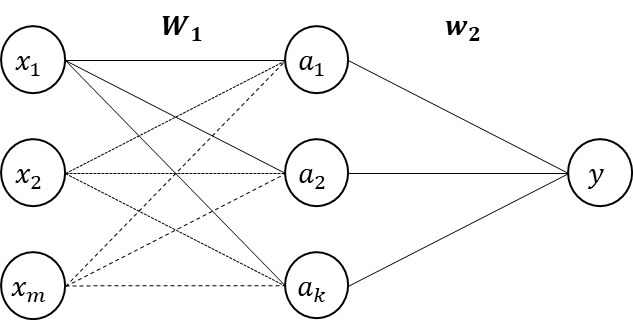


Figure 3: neural network with multiple neurons in hidden layer, source: author, Witzany (2017)

If the response variable *y* is continuous, the loss function used for optimization of the weights and biases would traditionally be the sum of squared residuals. Since in this thesis the point of concern is a binary dependent variable (event of default), further discussion goes straight into addressing the binary dependent variable problem.

Consider a binary response variable, the neural network for that optimization problem would have 2 output nodes, *y1* and *y2*, together forming a vector – **y**T = [*y1, y2*]T. Assume that *y1* would contain the raw output for the event occurring (i.e. attain value 1) and, conversely, that *y2* would contain the raw output value for the event not occurring (i.e. represented by value 0). The first question is (1) how to interpret the raw output values and the second question is (2) how to optimize the weights and biases of the neural network. The interpretation problem can be resolved by using the so called argmax function. It simply states that the chosen response corresponds to the *yp* node that has the highest raw value. However, although the argmax function provides a simple interpretation of the raw output values, it cannot be used for the mentioned optimization. The reason for this is that the slope of this function is always 0 at any point. To tackle the optimization problem, one can use the softmax function, which has the following form (James et al., 2017).

|  |  |
| --- | --- |
| , | # |

where is the exponent of *i*-th raw output divided by the denominator which contains the sum of exponents of all *p* raw outputs. In a binary classification problem, *p* equals 2. This provides a differentiable function and its *p* softmax calculations sum to 1. They are therefore interpreted as probabilities. Now, the softmax-transformed raw output values of the neural network are passed onto the cross-entropy function (James et al., 2017):

|  |  |
| --- | --- |
| , | # |

where the subscript *n* refers to the cross-entropy calculation of the *n*-th observation of the matrix of input features **X**. The *N* calculations are summed which generalizes the expression above into (James et al., 2017):

|  |  |
| --- | --- |
| . | # |

To optimize the neural network, the goal is to minimize the cross-entropy it outputs. The minimization problem is not solved analytically but rather an algorithm is chosen to solve the task. In this text, gradient descent is explained, and its steps are the following (James et al., 2017):

1. Randomly initialize the parameters (weights and biases) of the neural network.
2. Analytically express the derivatives of the loss function (cross-entropy) w.r.t. the parameters (weights and biases), this is the gradient of the loss function (Witzany, 2017):

|  |  |
| --- | --- |
| . | # |

Note that the negative of the gradient is taken in order to move in the opposite direction. The derivatives w.r.t. the respective weight parameters can be expressed using the chain rule.

1. Use the parameters generated at point (1) as inputs for the analytical forms in step (2), save the result.
2. Multiply the result from (3) with a learning rate and save the result.
3. Subtract the result from (4) from the initialized values of weights and biases.
4. Use (5) as the new weights and biases and repeat the process (steps 3-6) until some arbitrary criterion is met. The criterion is set so the gradient would be approximately 0.

### Support Vector Machines

The basic idea of the Support Vector Machines (SVM) model starts from the Maximal Margin Classifier (MMC). The latter fits a hyperplane to data which is linearly separable. The MMCs optimization problem is defined as (James et al., 2017):

|  |  |
| --- | --- |
| , | # |

subject to:

|  |  |
| --- | --- |
| , | # |

and,

|  |  |
| --- | --- |
|  | # |

where *M* is the margin area around the hyperplane. In this basic MMC model, no data points are allowed to be in the marginal area. This is commonly referred to as the *hard margin*. The goal of the maximization problem is to fit a hyperplane such that it correctly classifies the observed data points and maintains the *M*-distance of all data points from the hyperplane itself. Correctly classifying the data with a linear hyperplane (including a margin) strictly means that no errors can be made. In practice this ends up narrowing the margin. However, what is more frequent in practice is also that observations cannot be separated with a hyperplane and some error tolerance should exits. The Support Vector Classifier (SVC) relaxes the hard margin condition and introduces a *soft margin*, the optimization problem is re-defined as (James et al., 2017):

|  |  |
| --- | --- |
|  | # |

subject to:

|  |  |
| --- | --- |
| , | # |

and,

|  |  |
| --- | --- |
|  | # |

where:

|  |  |
| --- | --- |
| and . | # |

The variable is known as the *slack variable* and, depending on its value, it will allow its incorrect classification. The *tuning parameter* *C* introduces the possibility to control the slack variable. The larger the tuning parameter, the larger the error tolerance of the classifier (James et al., 2017).

The decision boundaries that are created with the hard and soft margin algorithms described above are linear. Such a classifier cannot handle non-linear relationships between the features and the response variable. This is in some cases crucial for having an algorithm that can serve its purpose. To handle nonlinearities, it may suffice to scale the data, e.g. extend the hyperplane above by adding second powers of features. This would ensure a non-linear shape of the data-dividing hyperplane. A non-linearity is simply introduced into the margin condition, for example second powers addition (James et al., 2017):

|  |  |
| --- | --- |
|  | # |

The SVM approach is to express non-linear function such as the one written above through a so-called kernel function *K*, the which can be specified in different ways. For example, the polynomial kernel is (James et al., 2017):

|  |  |
| --- | --- |
| , | # |

where *d* stands for degree. Another popular choice is the radial kernel (James et al., 2017):

|  |  |
| --- | --- |
| , | # |

where γ is larger than 0. Whichever *K* is chosen, it is always about the evaluation of dot products of pairs of observations. This kernel function is then embedded into the support vector classifier (James et al., 2017):

|  |  |
| --- | --- |
| . | # |

### Tree-based Methods

The building block of the random forest algorithm is a single decision tree. In the case of this thesis, the classification tree is relevant. The classification tree passes observations onto a sequence of questions. It is designed in such way that the evaluation of a metric *i* (a binary “yes” or “no”) determines what the *i+1* metric will be. This results in branching – for example, at initialization there is only 1 classification metric (node) with 2 evaluations; This results in 2 new nodes with 4 evaluations, and so on until the *leaf*/*terminal* nodes. In practice classification trees do not need to grow symmetrically. Stopping criteria, “pruning” techniques or simply achieving “purity” of the terminal node might result in an asymmetric tree (James et al., 2017). An asymmetric decision tree is depicted schematically on the following figure.

Figure.

To optimize for an adequate tree, one looks at (1) the optimal sequence of binary classifications based on the discriminatory power of individual features (**X**) and (2) the termination criteria for the generation of new nodes. The discriminatory power of individual features can be evaluated using different criteria. For example, the Gini coefficient of discrete variables is (James et al., 2017):

|  |  |
| --- | --- |
| , | # |

where is the probability that the class *k* will be obtained in the *m-th* region (node). If the probability of observing *k* is high, then *G* will be relatively low. This indicates high discriminatory power. Another popular metric is entropy (James et al., 2017):

|  |  |
| --- | --- |
| , | # |

which also takes upon relatively low values for highly discriminating regions of observations. For example, a dataset may contain information about whether credit card applicants had a permanent employment contract at the moment of applying. Also, assume that this dataset has information about default events. The more often the “no permanent job contract” classification coincides with the “defaulted” classification in the dependent variable as well as the opposite holding for the applicants with a permanent job contract, the lower the *G* and *D* will be. Calculating *G* and *D* is more involved in the case of continuous explanatory variables. There, a continuous variable is first sorted. Then, averages of 2 nearest observations are calculated. These points are candidates for the discretization of the continuous variable. The best performing threshold as calculated by Gini or entropy is then used as a data transformation for the classification tree. A single classification tree is not robust and it typically overfits the training data. This issue is tackled by bagging, random forests and boosting methods (James et al., 2017).

#### Bagging

In classification problems, bagging simply refers to (James et al., 2017):

1. estimating multiple decisions trees on subsets of the training data observations (this data can be bootstrapped in the case there is not enough data),
2. collect the predicted classification for each observation and take the most frequent classification of each observation as the final classification.

Once many outcomes are averaged in this way, the simple interpretability of a decision tree is lost. The *variable importance* measure is a way of reconciling the contribution of individual features to the overall performance of the model. In the case of classification problems, tracked are contributions of individual variables to the overall Gini (James et al., 2017).

#### Random Forests

Random forests extend the idea of bagging by not only limiting the training data to subsets of observations, but also to subsets of explanatory variables. Consider a number of features *m*, then the random forest algorithm randomly chooses *p* features such that *p* ≤ *m*. The smaller the *p*, the more different the approach is from bagging. Conversely, if *p* = *m*, the random forest would practically yield the same result as bagging. The goal of the random forest approach is to have less correlated decision trees (James et al., 2017).

#### Boosting

The basic idea behind boosting methods is to build consecutive trees. Each tree learns from the errors made by the previous ones. Popular are AdaBoost and gradient boosting, there are also many other variations.

AdaBoost creates simple decisions trees with one metric and 2 possible results. Such trees are called *stumps* amongst practitioners.

1. At initialization, each observation is assigned a weight of 1/*N*.
2. The errors of a former stump are stored and are used as input for the next one.
3. The observations on which the former stump predicts incorrectly are assigned with a new weight which is larger relative to the correctly predicted ones.
4. A new dataset is generated where weights are used as inputs. This results in a synthetic repetition of the incorrectly predicted observations. The next stump is therefore more sensitive to these.

Steps 2-4 are repeated until some stopping criteria is met.

Gradient boosting has its name given the small steps approach based on the learning rate.

1. At initialization of a classification gradient boost algorithm, each observation is assigned with the overall log-odds of the dataset. These are then transformed to probabilities using the sigmoid function.
2. The pseudo-residuals of the former step are obtained as differences between the predicted probabilities and the *observed probabilities* (which are 0 and 1). They are used as inputs for the next step, which is the creation of a new decision tree.
3. The next decision tree forecasts the residuals of the previous step. Their terminal leaves contain values of those residuals. Now, their output values have to be transformed, for example for one leaf by:

|  |  |
| --- | --- |
| , | # |

where *i* is the current decision tree, *i-1* is the previous step and *K* is the number of observations in one leaf. This is repeated for every leaf of the decision tree *i*.

1. The raw output value from the previous step is multiplied by the learning rate and added to the previous steps to obtain the log-odds *i*.
2. The log-odds *i* is transformed into a probability for example using the sigmoid function.

Steps 2-5 are repeated until some stopping criteria is met.

### KNN

Consider a dataset of known outcomes **y** and corresponding explanatory features **X**. For a new observation *yi* and **xi**, the nearest neighbor approach calculates the fitted value as the average of *K* nearest observations (Witzany, 2017):

|  |  |
| --- | --- |
| . | # |

The formulation above does not contain the right-hand side features. In fact, the explanatory variables are used only to identify the nearest observations. Hence, the main problematic of this method surrounds the question of how to identify the most similar ones. The most straightforward way to identify similar observations is to use the Euclidian distance (Witzany, 2017):

|  |  |
| --- | --- |
| . | # |

Categorical variables have to be converted into numerical in order to enable this computation (Witzany, 2017). This can be achieved by, for example, introducing a dummy variable transformation or calculating the Weight of Evidence.

## Model Testing

In this section, various model accuracy testing approaches are reviewed.

### Accuracy Ratio

Consider a non-defaulted *good* debtor G and a defaulted *bad* debtor B. It is then considered that a rating tool performs well w.r.t these 2 debtors if the rating of the debtor G is higher than the rating of the debtor B. In total, there are three possible scenarios and each can be assigned a probability (Witzany, 2017):

|  |  |
| --- | --- |
| , | # |
| , and | # |
| . | # |

The accuracy ratio can be expressed in terms of probabilities (Witzany, 2017):

|  |  |
| --- | --- |
|  | # |

The accuracy ratio is a summary statistic of the cumulative accuracy profile (CAP) (Engelmann et al., 2017). Consider 3 random variables – *ST*, *SD* and *SND* which denote score distributions of all, defaulted and non-defaulted debtors respectively. Engelmann et al. (2017) then define that the probability that a debtor *j* will receive a score value of *i* is:

|  |  |
| --- | --- |
| , | # |

where π is the probability of default and the total score of the debtor at hand is the sum of the defaulted and non-defaulted score probabilities – *pD* and *pND*. The cumulative probabilities *CDD*, *CDND* and *CDT* are simply defined as sums of probabilities of different scores *i*.

Continue.

### Area Under the Curve

The Area Under the Curve (AUC) is defined as (Witzany, 2017):

|  |  |
| --- | --- |
|  | # |

where the assignment of equal ratings to *G* and *B* is taken as a half-success. Witzany (2017) further notes that the AUC is a linear transformation of AR.

Confusion matrix, , KS Distance

## Data Preprocessing

### Oversampling

In cases when a classification problem is subject to a large disbalance between its classes, the estimated models might not be able to discriminate between classes. In credit risk datasets, the number of bad observations is always lower than the number of good observations. Consequently, a classifier that cannot handle unbalanced data would classify all observations into the larger class. Such a model is obviously useless.

Oversampling techniques are introduced to solve this issue. These techniques address the class imbalance problem by synthetically creating data. Popular are variations of the Synthetic Minority Over-sampling Technique (SMOTE) – BorderlineSMOTE, KMeansSMOTE, SVMSMOTE.

The basic SMOTE algorithm utilizes the idea of the KNN solution. A difference between, for example, 2 observations of the underrepresented sample is computed. Then, the difference is multiplied with a random number between 0 and 1. This exercise is repeated for a number of times that is sufficient to balance-out the classes in the dataset. More specifically, a new observation of 2 nearest neighbors is generated by the following formulae:

|  |  |
| --- | --- |
|  | # |

where attains values between 0 and 1, and , are actual observations from the minority class.

Another popularly used approach is ADASYN. It is conceptually the same as SMOTE. It practically extends the SMOTE algorithm by adding a random term to the generated observations. Therefore, the generated observation is no longer a strict linear combination of the neighbors.

|  |  |
| --- | --- |
|  | # |

where is the random term.

### Data Transformations

Binning, WoE. Variable selection

### Forward feature selection

Text.

# Application on Czech Mortgages Portfolio

The modeling dataset concerns a part of the Czech mortgages portfolio of a major bank in the country. The available dataset has observations for which at least 6 months elapsed since the moment of contract signing. Therefore, the approach to scoring will be the behavioral one. Given a large number of columns in the dataset – 325, an automated variable selection based on statistical metrics is performed. Only after shortlisting the explanatory variables, a more granular analysis of the features is performed. The rest of the section is organized as follows:

The dataset has 200,000 observations, out of which 726 facilities were marked as defaulted. This amounts to an overall default rate of 0.36%.

Table 2. Overview of the modeling dataset

|  |  |  |  |
| --- | --- | --- | --- |
| cohort date | # observations | # defaults | default rate (%) |
| 201101 | 15,182 | 103 | 0.68% |
| 201201 | 16,230 | 82 | 0.51% |
| 201301 | 18,936 | 104 | 0.55% |
| 201401 | 20,153 | 134 | 0.66% |
| 201501 | 21,923 | 73 | 0.33% |
| 201601 | 23,727 | 84 | 0.35% |
| 201701 | 25,930 | 70 | 0.27% |
| 201801 | 28,406 | 36 | 0.13% |
| 201901 | 29,513 | 40 | 0.14% |

Source: CSOB.

Without suggesting the possible drivers of the development of the portfolio as outlined in the table above, it can be observed that the default rate showed a downward trend in both absolute and relative terms. The analyzed mortgage portfolio doubled in size in less than 10 years, both in terms of the number of facilities as well as the outstanding amount which increased from 1,600bn to 3,700bn CZK over the period.

## Modeling

In this section, the preprocessing, exploratory data analysis and modeling steps are outlined.

### Preprocessing: data cleaning

The focus of the data preprocessing procedure is on getting the data into an analyzable shape from the technical point of view. The longlist of all available variables along with their descriptions is included in Appendix X. The treatment of missing values is performed depending on the characteristics of a variable. For example, a nan value in the case of a delinquency flag column implies that no delinquency took place. On the other hand, practically nothing can be reliably assumed about a missing outstanding amount. The rates of missing entries per column in the dataset as well as their treatment is also included in Appendix X. Globally, the following decisions are made:

1. Missing entries of variables of string type such as topographical names, academic titles etc. are not treated. Their categorical derivatives are however subject to decision-making. For example, the worst category can be assigned to missing cells.
2. The missing integer entries of variables that code the delinquency status are replaced with 0 values. This is done based on the implication that no delinquency happened.
3. Other integer variables are not treated for missing entries.
4. Count variables, such as number of months are not treated for missing entries.
5. Missing entries of flag variables are resolved by inputting 0 values.

A more refined and specific treatment of some variables that exhibit missing amounts might be taken into consideration upon their inclusion into a shortlist. The scope of explanatory variables that enter a shortlist depends on univariate analyses. Since each estimation techniques as explained in Section X are preceded with different univariate analyses, each feature selection (shortlist) might also look different. Aside from the missing values, the data quality is sufficiently good to proceed with further analyses.

### EDA

Text.

### Preprocessing: transformations and feature selection

In this subsection, the transformative and feature selection data preprocessing steps are outlined.

The data is split into training and testing samples in the 70:30 ratio. For replicability of the results, the seed is set to 130816. Two types of datasets are going to be tried in each model estimation:

1. A preprocessed dataset without oversampling, and
2. A preprocessed dataset with oversampling.

The intention of using these two versions of datasets is to capture how well individual estimation techniques perform in the case when the data is class-imbalanced and when it is not.

#### Oversampling

The chosen oversampling technique is SMOT. This solution is chosen over the more refined ADASYNC oversampling technique due to the fact that the available software solution for SMOT in Python is able to elegantly handle categorical variables. The Python library used in this application is imblearn, the module over\_sampling has the function SMOTENC (the “ENC” part of its name representing the ability to encode categorical variables). Five nearest neighbors are linearly combined in order to arrive at each new observation. The choice of 5 neighbors is rather arbitrary as it is the default setting in the SMOTENC implementation.

#### Binning and WoE

Now, the outlier treatment, binning and the WoE transformation is computed by one function. The Python library scorecardpy has a complete infrastructure for probability of default modeling in credit risk applications. In this instance, the function woebin is used. It performs binning and subsequently computes the WoE values. Also, outliers are taken out when bins are being defined. The relatively straightforward interquartile range method is used to address these. Then, the bins are applied back to the whole dataset (including the outliers) using the function woebin\_ply.

Since the longlist of variables is fairly large, it would not be economical to plot and inspect the binning and the resulting WoE transformation for all explanatory variables. Therefore, only several variables are shown and their economic interpretation is outlined. In the Czech mortgages dataset the retail behavioral score of clients is collected. This refers to the behavior of clients on their retail non-mortgage products that they have at the bank. The higher the score, the lower the default rate should be. The 2 plots below show that this simple economic expectation is fulfilled – the WoE values drop monotonically with lower score bins.

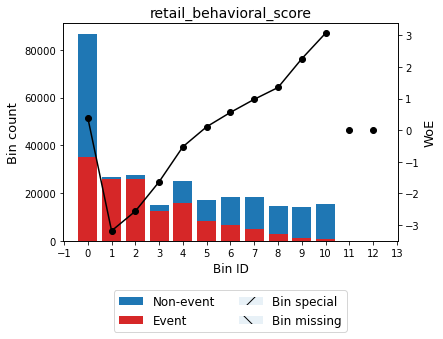


Figure 4: binning and WoE transformation for retail behavioral score, source: author

The WoE values across bins drop monotonically going from left to right in the case of the number of days in delinquency. This is also expected from sanity check standpoint. The longer the client struggles to meet their obligations, the higher the chance that the financial issues they encounter are substantial. This concretely is a 6M average of the count of days in delinquency. In the dataset, other averages are available for this as well as the simple count of days in delinquency as at the snapshot moment. From the plot, it can also be observed that most clients did not ever become delinquent.

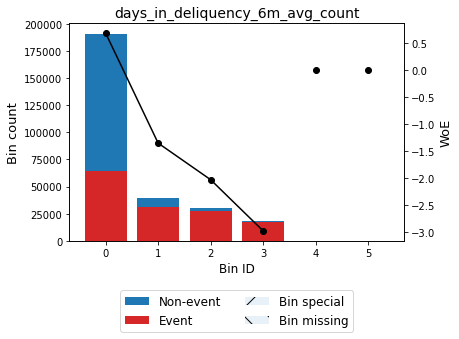


Figure 5: binning and WoE transformation for 6M-averaged count of days in deliquency, source: author

The low loan-to-value ratio (LTV) indicates clients that had substantial share of their own funds to participate in the purchase of the real estate asset, relative to its value. A sharp drop in the WoE follows to clients that had less cash available at the moment of purchase. Finally, the WoE-values increase that follows after that group could be explained only in interaction with other metrics. For example, these clients might have had higher retail scores, disposable collaterals, guarantors or other reasons for which they were granted with a high LTV in the first place. Ultimately, it appears that these clients did not struggle to manage the large loans as much as the group in the middle of the plot. It can be concluded that the LTV alone is not a metric that can explain the future performance of a client.

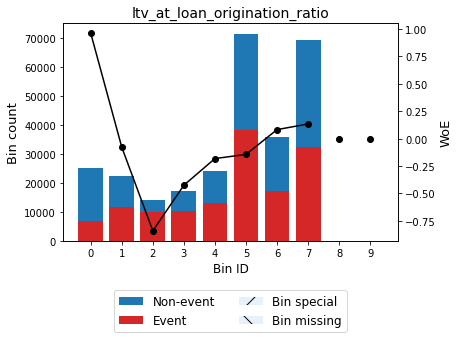


Figure 6: binning and WoE for loan-to-value at origination ratio, source: author

#### Univariate Analysis

Now, the univariate analysis can either be performed in the same way for all subsequent multivariate models. The second option is to perform them in such way that matches the method of the final model, if possible. For example the multivariate logistic regression to be preceded with a battery of univariate logistic regressions. In some cases this would not be possible, for example estimating a univariate neural network would be a questionable and time-consuming exercise. Finally, one can consider skipping the feature selection step based on the univariate models. In that case, some features would be excluded based on the multicollinearity assessment. Ultimately, the final set of features would be arrived at by limiting the maximum number of explanatory variables in the estimated model. The explanatory variables that would stay in the model are the ones that perform the best.

In this work, it is chosen to perform the univariate feature selection based on traditional credit risk practices. The aim of this feature selection procedure is not to arrive at the final set of explanatory variables. Rather, only features that clearly underperform are excluded at this instance. Two criteria are used and both have to be satisfied – the IV value has to be above 0.02 and the gini coefficient from a univariate logistic regression has to be above 0.1. The univariate feature exclusion procedures are ran separately on both the initial and SMOT-extended datasets.

#### Multicollinearity Removal

Finally, the multicollinearity assessment on the remaining variables is performed using the SelectNonCollinear function from the collinearity library. The maximum tolerated correlation between 2 explanatory variables is 0.5 (threshold set as per Witzany YYYY). When there is a group of variables with correlation above the specified threshold, the one that has the strongest ANOVA-F statistic with respect to the target variable is chosen.

### Logistic regression

In this subsection, the logistic regression model is applied. The number of remaining explanatory variables is rather high and they should be reduced to some 7-15 as per Witzany (yyyy). To fulfill this optimization task, the forward selection algorithm is used. The LogisticRegression function from the linear\_model module of the sklearn library is combined with the SequentialFeatureSelector function from the feature\_selection module of the mlxtend library. Finally, the explanatory variables used in the construction of the logistic regression model are listed in the table below.

Table 3. Variable selection using the forward selection algorithm

|  |  |
| --- | --- |
| variable\_name | IV |
| age |  |
| brki\_installment\_amt |  |
| collateral\_required\_amt |  |
| days\_in\_deliquency\_6m\_max\_to\_next\_installment\_ratio |  |
| debt\_summary\_2qs\_max\_amt |  |
| education\_categorical |  |
| interest\_paid\_to\_next\_installment\_6m\_max\_ratio |  |
| penalty\_interest\_paid\_mtd\_amt |  |
| product\_type |  |
| retail\_behavioral\_score |  |

Source: author

The AUC is similar and indicates good model performance on both the training and the test samples. The D-value of the Kolmogorov-Smirnov statistics are also similar across the train and test samples.

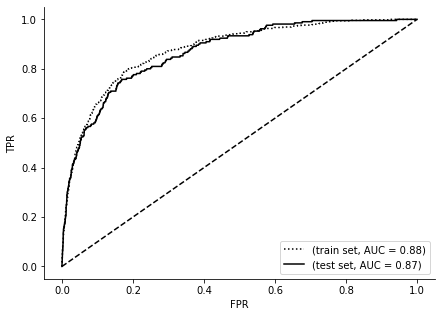


Figure 7: ROC cuve for train and test sets, logistic regression, source: author

Add confusion matrix.

### Neural Network

The sklearn library implemented in Python is the programming basis of this computation. There, a multilayer perceptron classifier solution is available. The most troublesome part of a neural network estimation is narrowing down to the best one given the vast options (number of hidden layers, activation function, solver algorithm, learning rate, regularization term). In order to tackle the search for the most appropriate neural network, multiple combinations are tried out and the best fitting neural network is selected. All combinations of the settings listed in the table below are considered. The same list of variables as in the logistic regression section is used to estimate this model.

Table 4: combinations for optimal neural network search

| parameter | combination |
| --- | --- |
| Hidden layer sizes *(the number of integers in the bracket implies the number of layers, n integers means n layers)* | [7] |
| [7, 7] |
| [10, 5] |
| [5, 5] |
| Activation function | Logistic |
| Tanh |
| Relu |
| Solver | Sgd |
| Adam |
| Alpha | 0.001 |
| Learning rate | Constant |
| Invscaling |
| Adaptive |

Source: author

The structure of the resulting neural network is summarized below.

Table 5: best performing neural network

|  |  |
| --- | --- |
| parameter | combination |
| Hidden layer sizes | [7] |
| Activation function | Logistic |
| Solver | Adam |
| Alpha | 0.001 |
| Learning rate | Constant |

Source: author

It’s performance on the train and test samples is mutually consistent.

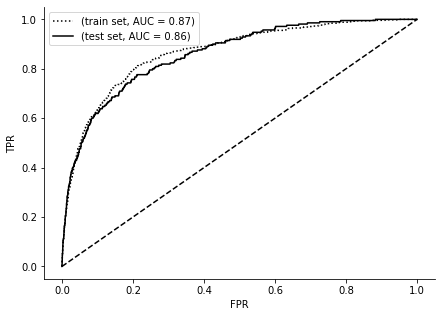


Figure 8: ROC curve for train and tests sets, artificial neural network, source: author

Next, a neural network using all explanatory variables is estimated. A larger number of neurons was introduced into the structure in order to accommodate 44 features. The optimal neural network in this case has a single hidden layer of 22 neurons, the activation function is sigmoid, it has a constant learning rate and it is solved by the adam algorithm. As can be seen from the performance assessment below, this neural network has a marginally better ROC curve for the train and test sets (reference figure). Therefore, it can be concluded that using the brute force of the large number of variables rather brings complications in the area of model management and implementation.

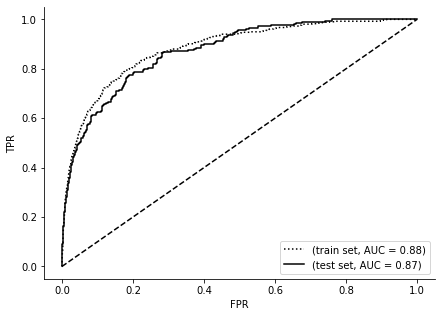


Figure 9: ROC curve for train and test sets, artificial neural network with all available variables, source: author

## Comparison of Results

# Final Remarks

# References

Witzany, J. (2017). *Credit Risk Management*. Springer Cham. doi: 10.1007/978-3-319-49800-3

Pech, C. (2016). *CS109 Intro to Probability for Computer Scientists: Handout #40*. Stanford University.

James, G., Witten, D., Hastie, T., Tibshirani, R. & Taylor, J. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer Cham. doi: 10.1007/978-3-031-38747-0

# Graphics and Tables

Graphics

**No table of figures entries found.**

Tables

**No table of figures entries found.**

# Appendices

Appendix A –

Appendix B –

Appendix A – Title

Appendix B – Title