**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](#_Toc164975616)

[1 Institutional Framework and Current Status 7](#_Toc164975617)

[1.1 Regulation: EBA Note 7](#_Toc164975618)

[1.1.1 Use of ML in IRB 7](#_Toc164975619)

[1.1.2 Complexity of ML 8](#_Toc164975620)

[1.1.3 Tracking Model Changes 10](#_Toc164975621)

[1.1.4 GDPR and AI Acts 10](#_Toc164975622)

[1.1.5 EBA Recommendations 11](#_Toc164975623)

[1.2 Literature Review 12](#_Toc164975624)

[2 Overview of Modeling Methodology 16](#_Toc164975625)

[2.1 Model Estimation 16](#_Toc164975626)

[2.1.1 Logistic regression 16](#_Toc164975627)

[2.1.2 Neural Network 18](#_Toc164975628)

[2.1.3 Support Vector Machines 22](#_Toc164975629)

[2.1.4 Tree-based Methods 24](#_Toc164975630)

[2.1.5 KNN 27](#_Toc164975631)

[2.2 Model Testing 27](#_Toc164975632)

[2.2.1 Accuracy Ratio and CAP 27](#_Toc164975633)

[2.2.2 AUC and ROC 28](#_Toc164975634)

[2.2.3 Rates of Error 29](#_Toc164975635)

[2.2.4 Hosmer-Lemeshow Test 30](#_Toc164975636)

[2.2.5 Population Stability Index 30](#_Toc164975637)

[2.3 Preprocessing 31](#_Toc164975638)

[2.3.1 Feature selection criteria 31](#_Toc164975639)

[2.3.2 Data Transformations 32](#_Toc164975640)

[2.3.3 Reject inference 33](#_Toc164975641)

[3 Application to Czech Home Loans Portfolio 35](#_Toc164975642)

[3.1 Data 35](#_Toc164975643)

[3.2 Programming Solution 36](#_Toc164975644)

[3.2.1 Data Ingestion 36](#_Toc164975645)

[3.2.2 Preprocessing: transformations and shortlisting 37](#_Toc164975646)

[3.2.3 Modeling 37](#_Toc164975647)

[3.2.4 Validation 38](#_Toc164975648)

[3.3 Application 39](#_Toc164975649)

[3.3.1 Estimated Models 39](#_Toc164975650)

[3.3.2 Results 41](#_Toc164975651)

[3.3.3 Comparison 45](#_Toc164975652)

[3.3.4 Logistic regression 45](#_Toc164975653)

[3.3.5 Neural Network 46](#_Toc164975654)

[Final Remarks 49](#_Toc164975655)

[References 50](#_Toc164975656)

[Graphics and Tables 51](#_Toc164975657)

[Appendices 52](#_Toc164975658)

# Introduction

Text.

# Institutional Framework and Current Status

In this section, the current regulatory trends are reviewed. A general discussion about the IRB framework and the ongoing implementation of Basel IV is skipped. The aim is to keep the discussion specific to the issue of machine learning. In the second part of this section, a literature review is outlined.

## Regulation: EBA Note

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” (EBA, 2023). The document addresses the use of ML techniques by the industry, their complexity and their interaction with the GDPR Act (EU, 2016) and the EU AI Act (European Parliament, 2023). The ML techniques are understood as “*models characterized by a high number of parameters, that require a large volume of (potentially unstructured) data for their estimation and that are able to reflect non-linear relations between the variables.*”. When the term is used in this chapter, the same understanding is meant. The note itself is a reflection to a 2021 questionnaire EBA/DP/2021/04 issued by EBA (EBA, 2021). It contained 17 questions and 14 institutions answered them. The EBA notes that 8 of the answers were provided by associations. It remains unclear whether these eight answers provided by associations represent a combined view of multiple financial institutions that are members of these associations. In the rest of this chapter, individual sections of the analyzed note are being paraphrased and reviewed. The scope of the review is limited to uses and formulations relevant for PD modeling under IRB.

### Use of ML in IRB

The note addresses the following use areas:

1. Firstly, the terms risk differentiation and risk quantification as outlined in the EBA modeling guidelines (EBA, 2017) are pinpointed. The former refers to identifying the risk drivers that have satisfactory discriminatory power. The latter refers to calibrating the risk parameters in such a way that they reflect the long-run averages. It is reported that financial institutions use ML techniques for risk differentiation but not for calibration. The latter is namely not possible due to shorter time series of new data sources. This might imply that when ML techniques are used for modeling, a larger scale of features is considered, i.e. it is not just the estimation algorithm that changes.
2. It is noted that ML techniques are not always outright used for model estimation. They may be used for some supportive and model-enhancing operations such as text mining, missing data imputation and tackling of unstructured data.
3. It is noted that the added value of ML is especially noted in cases when risk differentiation was not successful. ML techniques are able to exploit non-linearities and thus find relationships that cannot be easily identified by more traditional estimation approaches.
4. It is noted that challenger models are often ML-based, whilst the tested models can remain traditional. In this way, the end model that is going to be used is still the interpretable and fully IRB-compliant traditional model. Nonetheless, using ML as the benchmark will always provide input into either how better the tested model might be or whether it should be replaced by an ML model. It is, however, also mentioned that challenging a traditional model with an ML model is not so straightforward. For example, if a validator wants to test the significance of individual features, this is easily done in a logistic regression-based model. The contribution of individual features in the case of a neural network is, however, much harder to pinpoint.

### Complexity of ML

The note documents that the industry practitioners reported issues in the areas of statistical inference, know-how and model interpretation.

1. The industry respondents reported overfitting of the training sample as the main issue of using ML techniques. One case is highlighted by which this is more pronounced in the case of low default portfolios. It is not, however, clear to what extent the low-default portfolio characteristic causes issues. For example, both home loans and municipal portfolios are low default portfolios in relative terms. The difference is although in the sizes of these 2 portfolios. Whilst there are only several hundred municipalities per mid-sized country and defaults are quite rare, there are hundreds of thousands of observations on home loan portfolios. A reflection on this point is to be made in the empirical part of this thesis. On a general note, it is well known that large datasets are required for ML models in order to prevent overfitting the training sample.
2. It is reported that one respondent had difficulties implementing standardized solutions to model development and model validation when ML techniques are used. The model development procedure should generally be streamlined and algorithmic. In the case of neural networks and other ML techniques, this is all that more important due to the variety of options to tune hyperparameters. Although an initial validation might require case-by-case individual tests, annual validations can be standardized with relative ease and ML implementations should not pose an issue. It is therefore concluded that the fact that only one respondent reported this as an issue is rather an outlier.
3. Financial institutions responded that additional know-how must be acquired in order to enable the usage of ML models. This implies that modeling departments in individual banks do not necessarily have an established group of experts in the area. In addition, once ML expertise is acquired in the form of new hires, it takes time to get them aligned with the understanding of the credit risk management ecosystem.
4. It is reported that the explainability of ML models is challenging. In its note, EBA (2023) collected responses about what banking practitioners use to explain an ML model:
   1. Shapley values (40%),
   2. model documentation (28%),
   3. graphical tools (20%), and
   4. sensitivity analysis (8%).

Although not explicitly said in the note, here it is understood that, for example, 40% of respondents use Shapley values. Conversely, 60% do not or they did not report it. It is also understood that the respondents use more than one metric outlined above.

1. EBA notes that respondents reported traceability as an issue. Nonetheless, a specific commentary on how financial institutions approach this is not provided.

### Tracking Model Changes

The view of the EBA (2023) is that the model change framework applies ML models as well. In particular, they repeat that a model change that results in a significant change in the rank ordering is a material model change. In addition, it is highlighted that a single material model change should not be artificially split into several immaterial model changes. The last point warrants a discussion about whether the current understanding of what makes a single model change fits into the ML workflow. It is noted that financial institutions require further clarification.

### GDPR and AI Acts

EBA (2023) notes that the respondents did not provide much specific commentary w.r.t. IRB-GDPR interaction. In the text it is therefore stated that GDPR explicitly prohibits the use of certain kinds of personal data for creditworthiness estimation. It is also prohibited to track information from social networks and use it as modeling input. Finally, the so-called *minimization principle* is to be followed. It is understood as “*adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed.*”. This means that banking institutions have to, if already not, tighten their management of such sensitive data. For example, sensitive information about retail portfolio individuals cannot be easily accessible by employees. Rather, a time-limited request for access can be reviewed upon sufficient use case argumentation and granted for example for 24h.

At the time of writing of this thesis the AI act is being finalized. By this point, it is expected that the act will define the creditworthiness assessment as a “high-risk” system. More specifically because the determine “*persons’ access to financial resources*”. The act is expected to be in power to AI models that underwent a substantial change or were newly designed after a set date of the act stepping in power. In effect this means that if an AI IRB model was designed historically, it would not be subject to the AI act until it has a significant model change. However, it is not clear whether the *material model change* will meet the definition of a *significant model change* once the act is in place. EBA (2023) argues that a distinction between creditworthiness assessment for the purpose of loan granting should be decoupled from the same kind of assessment for the purpose of capital calculation. They argue that the latter does not have an impact on natural persons.

### EBA Recommendations

EBA (2023) defines the following principle-based recommendations:

1. “*All relevant should have an appropriate level of knowledge of the model’s functioning*”. Some of these stakeholders are: model developers & model owner, model validators, the CRO, business representatives.
2. “*Avoid unnecessary complexity*”. Financial institutions should avoid introducing an excessive number of variables into their models. It is not explained what *excessive* is, i.e. how many variables this could be. It is further recommended that structured data should be preferred to unstructured data, when possible. Finally, it is argued that a simpler model should be chosen over a more complex one in the case that the performance is similar. This point can be addressed through automated model validation. For example, if a neural network is tested against an automated challenger logistic regression-based model and the performance difference is negligible, then the neural network should be abolished or re-designed.
3. Ensure that the “*model is correctly interpreted and understood*”. Financial institutions are recommended to analyze the impact of individual features, their contribution w.r.t. other features, have an economic understanding of the relationships in the model and provide a documentation of the points mentioned in this sentence. Attention should be paid to removing bias and overfitting. Although this requirement re-iterates the general process of modeling regardless of the kind of algorithm used, it explicitly strengthens the requirement around junction points that in practice might be avoided due to their complicated tackling.
4. Human judgement should be corroborated with appropriate understanding of the ML model that is being affected by this judgement.
5. It is recommended that a close eye is kept on regular model calibrations. The rationale is that credit risk should not be subject to frequent changes in the data generating process. At times, a structural shift in some key drivers would naturally lead to the necessity of re-calibrating the model, but this is not expected often.

## Literature Review

In this section, previously published research on the comparative analysis of different estimation techniques is reviewed. Since the literature on the matter is broad, the choice of the cited literature is limited to renowned publications as well as publications that are focused on applications on mortgage portfolios. To ensure reproducibility of the literature search, the following table summarizes the search criteria and the obtained results.

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 April 2014 to April 2024 | 340 | TBU,3 | https://www.webofscience.com/wos/woscc/summary/0cf5c555-09d9-4136-89f3-139c8f876da5-da7518dd/relevance/1 |
| Web of Science | 1. keywords: machine learning mortgage market; 2. filter: “4.61.1820 Credit Scoring” 3. filter: from April 2014 to April 2024 | 3 | 0 | https://www.webofscience.com/wos/woscc/summary/c8bf1a64-c3f2-4119-98bb-7c9991d4002c-da8dbe21/sort-group-background-citingcount/1 |
| Web of Science | 1. keywords: machine learning home loan; 2. filter: “4.61.1820 Credit Scoring” 3. filter: from April 2014 to April 2024 | 4 | 0 | https://www.webofscience.com/wos/woscc/summary/0748a1c4-abd4-4b06-b71b-6e67020c1984-da8ddb8a/sort-group-background-citingcount/1 |

Source: author’s search

The first search results in a large number of results. This is due to the fact that the used keywords can pick up on just anything that mentions ML w.r.t credit risk. Out of the four chosen articles, three represent frequently cited comparative studies of PD classification models and the last one is an extensive literature review. Finally, the search was repeated with keywords held specific to mortgage portfolios. The results were very narrow, and they did not particularly match the portfolio scope of this thesis. On top of the results obtained above, the first paper that is cited below was obtained rather randomly whilst looking for a most recent and reliable literature review kind of study.

A general view is provided by Shi et al. (2022) where they identify 76 published studies that have contributed to the field over the last years. The major findings that they report are that modern machine learning algorithms outperform the traditional statistical approaches. Secondly, they find that most papers reported ensemble methods as outperforming the machine learning algorithms. According to portfolio-type scoping, they split the reviewed studies into consumer and corporate finance portfolios. When researching the abstracts of the articles reported as consumer portfolios, it could not be confirmed that any of the studies tested models on mortgage portfolios. Rather, these would either be publicly available datasets or in several cases, undisclosed datasets procured from banks in Europe or elsewhere in the world, e.g. China.

Lessmann et al. (2015) perform extensive research testing 41 different approaches and 8 different datasets. In their literature review, they provide an overview of approximately 50 researches that were done by the time they wrote their article. They find that the datasets were on average relatively small. Many of them had around several hundred or thousand observations, usually less than 10,000. Out of the 8 datasets they use are from the retail segment, 3 exceed 10,000 observations where the largest dataset has 150,000 entries and a default rate of 6.7%. They refrain from using oversampling methods such as SMOTE as they argue that this is not of importance when algorithms are compared in relative terms. In addition, they are of the view that oversampling would mask potential sensitivities of some algorithms to class imbalance. They use 6 metrics to compare the estimated models – (1) the percentage correctly predicted (PCC) and the KS statistic for classification testing, (2) the area under the curve (AUC), the Gini index and the H-measure, and (3) the Brier score for calibration. Following the 41 approaches, they estimate and test 1,141 different models. They test a variety of individual classifiers which includes but is not limited to logistic regression, vanilla neural networks, support vector machines. They also look at classification models that stack underlying simpler models – random forests or boosted trees. Finally, they look at heterogenous ensemble learning. They find that various ensemble methods provide the best results. When looking at the specific individual classifiers, the best performance is found in artificial neural networks followed by the logistic regression. Interestingly, these 2 are the only individual learning models that tightly enter the top 20 (out of 41). Different bagging and boosting methods perform better than the previous 2. Nonetheless, the top spots are reserved for heterogenous ensemble learning methods, i.e. the methods that combine various types of models into one.

Barboza, Kimura and Altman (2017) published a study that is a 21st-century extension of the groundbreaking Altman (1968) paper. Unlike Lessmann et al. (2015), they investigate a narrower set of estimation methods and they have one dataset at their disposal. The techniques they consider are logistic regression, discriminant analysis, support vector machines, artificial neural networks, bagging, boosting and random forests. The modeling scope concerns the business segment. They use 10,000 observations about North American companies for the period of 1985-2005, 449 of which defaulted. They maintain class balance by randomly choosing 449 non-default companies. The compiled train dataset therefore has less than 1,000 observations. Balancing of classes via simple reducing the dominant class was performed in the 1968 study as well. There the split is 33 default and 33 non-default observations. The decision to create an artificially balanced sample set removes the ability of this study to assess how different methods deal with unbalanced sets. In addition, 449 default observations is substantially more than 33. This in fact should enable estimation algorithms to recognize patterns even though the observed default rate is just slightly lower than 4.5%. In the study, they further note that no standardization or data-transformative steps were made. Although they are aware that lacking data-transformative steps might reduce the power of some models, they opt to perform the comparison on data without this preprocessing. Removing major preprocessing steps such as data normalization or binning from the research makes the reported result on the performance of models less useful for the industry. A credit risk practitioner should be interested in a comparison of these models that was conducted in such a way that it reflects how the models would have been built in practice. The report that random forests, bagging, boosting and neural network models outperform the logistic regression approach. Versions of SVM as well as the linear discriminant analysis did not yield powerful models overall.

Another benchmark study was performed by Moscato et al. (2021). They ingest about 888 thousand observations from different peer-to-peer lending data sources. Despite having such a large number of observations, only 31 are marked as defaulted. Note that about 15 thousand observations are up to 120 days past due. Depending on the definition of default, some of these observations would in fact have been marked as default. Despite using a non-traditional portfolio which does not translate to the context of a banking institution due to having a heterogenous lender side, the study brings two things to light. Firstly, given the low number of registered defaults, synthetic data oversampling technique is employed. Interestingly, they found that the best performing model is a combination of logistic regression and random oversampling, which is the most naïve oversampling approach. Now, this is also dependent on the metric one gives most weight to. It can be observed that the best reported method does not always outperform other models by all means. Random forest appears to be the approach that is best equipped to deal with no-oversampling data.

# 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

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

### Logistic regression

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 the 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.

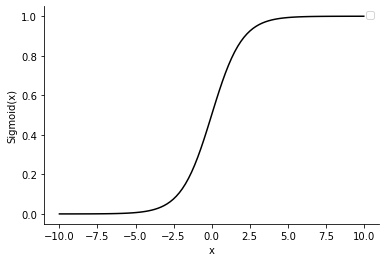


Figure 1: sigmoid function, source: author

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 toward 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*.

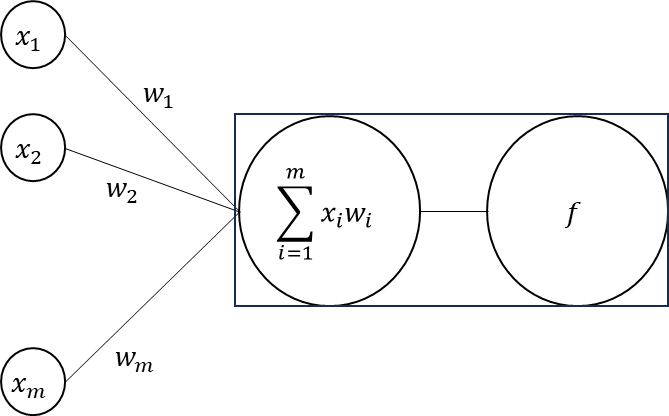


Figure : structure of a neuron, source: author, Witzany (2017)

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.

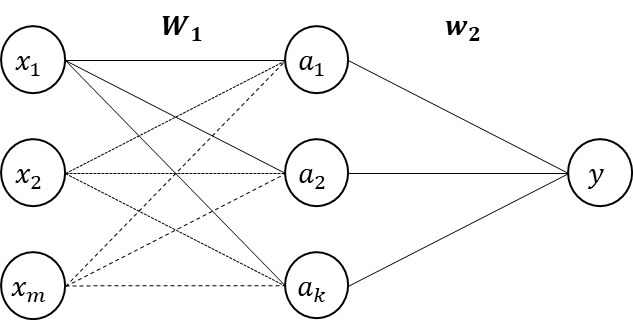


Figure : neural network with multiple neurons in hidden layer, source: author, Witzany (2017)

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 operational 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 handle a non-linear condition 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 and CAP

The accuracy ratio is a summary statistic of the cumulative accuracy profile (CAP) (Engelmann et al., 2003). Consider 3 random variables – *ST*, *SD* and *SND* which denote score distributions of all, defaulted and non-defaulted debtors respectively. Engelmann et al. (2003) 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*. Then, the CAP plot represents combinations of points *CDTi* and *CDDi* for every *i*. A random model would have all those points equal, which results in a straight line. A theoretical example of the CAP plot is outlined below.

Figure.

The larger the area between the CAP line and the random model line, the better the discriminatory power of the model. Conversely, the surface between the CAP line and the random model line can be divided by the surface between the perfect model line and the random model line. In that way, the accuracy ratio is obtained (Engelmann et al., 2003):

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

Witzany (2017) cites Engelmann et al. (2003) and further provides the following formulation. 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:

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

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

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

### AUC and ROC

The Area Under the Curve (AUC) is a summary statistic developed by using the Receiver Operating Characteristic (ROC) curve (Engelmann et al., 2003). As outlined in the section above, good and bad debtors have their associated distributions of possible scores that they can receive. Since rating models are imperfect in practice, one can expect an overlap between these 2 distributions as outlined in the figure below.

Figure.

Given the overlap, a cut-off value has to be introduced to rate a debtor as a good or bad. The optimization problem can be summarized in a confusion matrix.

Table : confusion matrix

|  |  |  |
| --- | --- | --- |
|  | default | non-default |
| below *C* | hit | false alarm |
| above *C* | miss | correct rejection |

Source: Engelmann et al., (2017)

In the confusion matrix above, the cut-off value is denoted as *C*. If the score is below *C* then the debtor is assigned with a default rating. The opposite holds. The 4 decisions made above can be summarized into several rates, Engelmann et al., (2003) are followed to define the hit and false alarm rates:

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

Where HR stands for hit rate and FAR stands for false alarm rate w.r.t. the defined cut-off value *C*, *S* stands for the assigned score. The metrics defined above are sufficient to construct the ROC curve – it is a plot of combinations of all HR(C*i*) and FAR(C*i*) for each *i*. The optimal cut-off value is the one that is the closest to the (0,1) point on the plot, i.e. the perfect model. Now, utilizing the whole ROC curve irrespective of the optimal cut-off value, the surface under it is calculated and the Area Under the Curve (AUC) is obtained. It is then divided by the surface of the perfect model. Engelmann et al., (2003) derive the probabilistic interpretation of the AUC. Witzany (2017) expresses the AUC in these terms:

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

where the assignment of equal ratings to *G* and *B* is taken as a half-success. Witzany (2017) as well as Engelmann et al., (2003) further note that the AUC is a linear transformation of AR:

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

### Rates of Error

Witzany (2017) describes the total weight of error (ER) as the weighted sum of Type I and Type II errors. In the confusion matrix outlined above, the Type I error corresponds to the false alarm, i.e. rejection of applicants that would have been performing. The Type II error corresponds to the miss rate, i.e. the approval of an applicant that subsequently defaults. The Type I and II errors are weighted by the proportions of good and bad applicants w.r.t. the total validation sample:

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

where HR(C) and FAR(C) are hit and false alarm rates as described in the section above given the cut-off C. πB and πG are probabilities of bad and good, respectively and they sum to 1.

The total rate of error introduced above is still somewhat impractical for business. Witzany (2017) notes that the impact of a single false positive is different than the impact of a single miss in financial terms. Therefore, the weighted cost of error (WCE) is:

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

where *l* is the loss realized from a bad debtor and *q* is the profit realized on a good debtor.

### Hosmer-Lemeshow Test

Witzany (2017) describes the Hosmer-Lemeshow (HL) which compares the predicted probabilities of default and realized default rates. The differences are squared and their sum is weighted:

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

where *s* refers to default grades, i.e. the observations have to be bucketed into default grades. The empirical default rates *ps* are then calculated per each bucket *s*. The H0 is that *PDs* is equal to *ps* for every *s*. The obtained statistic is compared against the chi-square distribution. Witzany (2017) further notes that the test is conservative due to empirically observable non-0 correlations between individual default events. This issue could be tackled by replacing the chi-square distribution through obtaining the empirical distribution of the statistic that takes these correlations into account.

### Population Stability Index

The Population Stability Index (PSI) conceptually makes most sense in the case of backtesting of already existing models (e.g. annual validations). Consider a model that was trained on data drawn from the period (*t*, *t+n*). Then consider that one more year passed and new data for the period *t+n+1* became available. The goal is to devise an index that would test whether the model still retains acceptable performance on the new data. The PSI is (Ernst & Young, 2021):

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

where *ps* is the observable default rate of the bucket s and *PDs* is the model default rate of the bucket *s*. The PSI metric can be produced while validating a model design as well. For example, the last fixed cohort can be used as the hold-out period against which the test would be computed. However, note that this would require training a model that does not use last cohort’s data, which would certainly not be the model that would be used in the very end. Splitting the data from cohorts into train-test would jeopardize the time comparison component. Nonetheless, consider a case in which 2 banks merge, such kind of a test would provide a way to compare the cross-applicability models they previously used. Many online texts and literature (for example: ) refer to these cut-offs as the rule of thumb:

1. PSI < 0.1: the tested model can still be used,
2. 0.1 <= PSI < 0.2: recalibration needed, and
3. PSI >= 0.2: redesign needed.

## Preprocessing

Thus far explained were different mathematical approaches to estimating a classification model as well as some general testing metrics of their performance. In this section, reviewed are major preprocessing operations. Here, data preprocessing is understood as any activity that comes before the actual mathematical calculation of the model and its subsequent testing.

### Feature selection criteria

Text.

#### Information Value (IV)

The WoE calculation explained above is also funneled into the assessment of explanatory power of features. Witzany (2017) defines the information value (IV) as:

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

which can be interpreted as the average WoE of a variable, where the weighting is done using the probability distributions. The value is always positive and the rule of thumb is to set the minimal performance threshold between 0.04 and 0.1. This means that the features not satisfying the criterion are to be excluded from further modeling.

#### Gini

#### Multicollinearity

#### Forward, Backward and Sequential

### Data Transformations

In this section, several data-transformative operations are reviewed.

#### Binning

Consider an explanatory feature **x**, which can be both continuous or discrete. Binning is an operation in which the *N* scalars of **x** are grouped into *S* buckets. The observations in bucket *s* usually obtain some common data-transformative treatment. For example, all entries of the bucket *s* can be replaced by their within-group average, or they can be transformed to represent the average within-group default rate (see WoE below). In any case, prior to applying the transformative treatment to the grouped data, the binning itself has to be performed.

In solving this thesis, the OptBinning library published Navas-Palencia (2020) is used. Navas-Palencia (2020) explains that the formulation of his binning procedure is globally comprised of 2 steps:

1. Initialization of bins that are (most probably) suboptimal. In the OptBinning programming solution, this can be performed via uniform/quantile discretization, with a decision tree or the minimum description length principle method (MDPL). The uniform/quantile discretization as well as the decision trees are implemented by Navas-Palencia (2020) using modules from the globally known Scikit-learn library (Pedregosa et al., 2011). The MDPL solution was programmed by Navas-Palencia (2020). When using the OptBinning solution, the default setting are decision trees. The splitting values as defined by the decision trees are used as the boundaries of buckets. In later sections of this thesis, the default setting will be followed.
2. Tuning of the initialized bins towards an optimal solution.

#### Weight of Evidence (WoE)

Witzany (2017) defines the weight of evidence (WoE) as:

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

where P[s|G] is the probability of a good debtor being observed in the bucket *s* and the same applies for bad debtors. In practice, the values obtained by the WoE transformation are often used to support the transformative treatment mentioned in the section above. Consider observations in a bucket *s* defined as per the section above. Then the WoE-transformation would be to replace all observations in the bucket *s* by the WoE value obtained for that bucket. If there are, for example, 5 bins, there would be a total of 5 WoE values. The WoE-transformed vectors are treated as continuous explanatory features.

### Reject inference

#### Undersampling

#### 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.

# Application to Czech Home Loans Portfolio

In this section, the methodology described in the previous chapters is put into an empirical review. Models are estimated using behavioral mortgage portfolio data of a major Czech bank. In the remainder of this section, the first subsection provides an overview of the method and hypotheses tested, the third subsection is a dataset review, the fourth subsection documents the preprocessing steps and finally, the fifth section outlines a comparison of different models.

## Data

The modeling dataset concerns a part of the mortgages portfolio of a major Czech bank. The available dataset has observations for which at least 6 months elapsed since the moment of contract signing. The dataset therefore enables the creation of a behavioral scorecard. The dataset has 200,000 observations, out of which 726 facilities were market as defaulted. This amounts to an overall observed default rate of 0.36%, which is typical for mortgage portfolios. As it is implied from the table below, the dataset was formatted in a fixed cohort setting. Re-structuring the dataset into flexible cohorts would not be possible.

Table 3. Overview of modeling dataset

|  |  |  |  |
| --- | --- | --- | --- |
| **cohort date** | **# observations** | **# defaults** | **default rate (%)** |
| 2011 | 15,182 | 103 | 0.68% |
| 2012 | 16,230 | 82 | 0.51% |
| 2013 | 18,936 | 104 | 0.55% |
| 2014 | 20,153 | 134 | 0.66% |
| 2015 | 21,923 | 73 | 0.33% |
| 2016 | 23,727 | 84 | 0.35% |
| 2017 | 25,930 | 70 | 0.27% |
| 2018 | 28,406 | 36 | 0.13% |
| 2019 | 29,513 | 40 | 0.14% |

Source: CSOB

It can be observed that whilst the number of facilities doubled by the end of the last decade, the portfolio experienced a decreasing number of defaults in absolute terms as well. The default rate is therefore 4 times lower than the one observable in the initial years.

## Programming Solution

The modeling exercise is performed in Python. Several Python classes are created and they are listed in the table below. The classes allow for a certain degree of variability in their settings. For example, the Preprocessor class can yield WoE-transformed columns as well as dummy-transformed columns.

Table 4: list of defined Python modules

|  |  |  |
| --- | --- | --- |
| Module name | Defined functions | Purpose |
| data\_getter | rename\_columns bool\_to\_flg exclude\_features ingest\_data | Basic wrangling of modeling dataset. |
| train\_test | split\_train\_test | Split into train and test sets. |
| oversampling | list\_categorical apply\_smotenc | Synthetic data oversampling. |
| woe\_binning | list\_categorical bin\_and\_woe\_transform | Binning and WoE transformation. |
| multicollinearity | remove\_multicollinearity | Multicollinearity removal. |
| logit | model\_logit | Logistic regression estimation and testing. |
| ann | model\_ann | Artificial neural network estimation and testing. |

Source: author

### Data Ingestion

The DataGetter class solves data manipulation issues that are specific to the dataset at hand. The longlist of all available variables along with their descriptions is included in Appendix X.

The treatment of missing values is performed depending on how the dataset is to be used upon having it passed to the Preprocessor class described in the next section. In the case that the intention is to use binning, minimal data manipulation operations are performed. These boil down to renaming via a mapping table, type changing, solving of minor encoding issues and feature exclusions based on a manually constructed mapping table. Missing entries are not treated proactively. They are left to the binning algorithm in the Preprocessor class. There, missing entries are assigned into a separate bin. In the case that binning is not performed, a more proactive treatment takes place. The following steps are implemented:

1. Missing entries of delinquency-related columns are imputed with 0 values in the case that there is no delinquency.
2. Missing entries of count and binary features are imputed with 0.
3. Features that have a large number of missing entries, such as ¼, and at the same time cannot be enriched in a less-involved manner are removed. Such variables are various types of amount entries.

### Preprocessing: transformations and shortlisting

The Preprocessor class performs operations on data that is already tidied up with the DataGetter class. The only data deficiency that it can deal with are missing entries, and that provided that they are taken along into a binning procedure.

In this class, several functions are defined and they solve the following tasks:

1. train-test split.
2. binning and transformation,
3. dummy variable creation, and
4. multicollinearity removal.

The train-test split function is straightforward. The binning and transformation function firstly performs the binning procedure as described in section XX. Then, features can be transformed into WoE based on the corresponding bins. Another option is to map bin IDs to features themselves and then have them transformed into dummy variables using the function (3) above. The multicollinearity removal function identifies groups of correlated variables and retains the strongest feature of that correlated group.

It is upon the user of the class to specify which of the functions are to be ran. For example, one reasonable combination might be to use (1), (2) and (3), where (2) is specified to end with a WoE transformation.

### Modeling

The Modeler class performs operations that are limited to feature selection, hyperparameter optimization and model estimation. The class has the following functionalities:

1. sequential feature selection,
2. hyperparameter grid search,
3. logistic regression estimation,
4. ANN estimation,
5. KNN estimation,
6. SVM estimation, and
7. RF estimation.

All of the functionalities above are built on primarily Scikit-learn classes (Pedregosa et al., 2011). In the case of feature selection, the selection metric, final number of features, forward/backward approach and the cross-validation parameters can be chosen. The hyperparameter grid search is always to be specified w.r.t. the relevant hyperparameters of the model that is being used. The output of the remaining functionalities is straightforward. For completeness, a list of adjustable hyperparameters/specific comments is provided:

1. hyperparameter optimization in the case of logistic regression is enabled only for L1/L2 penalizations and their combinations (elastic net); In the case that multicollinearity is being removed when using the Preprocessor class, the logistic regression functionality will not have any such penalization,
2. the ANN hyperparameter optimization takes in various combinations of hidden layer number and sizes, activation functions, solvers and learning rates. Early stopping is always activated,
3. the KNN hyperparameter optimization tries out a range of values for *K*, tests uniform and distance-based weighting of neighbors, Manhattan and the Euclidean distances,
4. the SVM hyperparameter search tries out a range of kernels, degrees of polynomials and regularization parameters *C*,
5. The RF hyperparameter search,

### Validation

The Validator class takes in train and test data as well as the model fits outputted from the Modeler class. It has the following functionalities:

1. prediction,
2. ROC plot creation,
3. CAP plot creation, and
4. K-S plot creation.

## Application

In this section, the preprocessing & modeling applications are listed. The preprocessing results are not commented in a systematical manner whilst the results of the modeling applications are taken upon more comprehensively.

### Estimated Models

The following combinations of data preprocessing and model estimations are considered.

Table : estimated models

| **IV & Gini shortlisting** | **Undersampling** | **Binning & WoE/dummy** | **Multicollinearity removal** | **Forward feature selection** | **Estimation techniques** |
| --- | --- | --- | --- | --- | --- |
| Yes | No | WoE | Yes | Yes | LR  ANN  KNN  SVM  RF |
| Yes | No | WoE | No | No | LR  ANN  KNN  SVM  RF |
| Yes | No | Dummy | Yes | No | LR  ANN  KNN  SVM  RF |
| Yes | No | Dummy | No | No | LR  ANN  KNN  SVM  RF |
| Yes | Yes | WoE | Yes | Yes | LR  ANN  KNN  SVM  RF |
| Yes | Yes | WoE | No | No | LR  ANN  KNN  SVM  RF |
| Yes | Yes | Dummy | Yes | No | LR  ANN  KNN  SVM  RF |
| Yes | Yes | Dummy | No | No | LR  ANN  KNN  SVM  RF |

Source: author

The initial shortlisting based on minimum Gini and IV criteria is always performed. One batch of models is trained on datasets that are not undersampled. Another 4 batches of models are trained on an undersampled dataset where the majority class is reduced in the number of observations. Then a regular non-undersampled test set where the majority class maintains its proportion is used for testing. Either WoE or dummy-based data transformations are used for model training. Multicollinearity is removed in half of the models and in the other half it is retained. The goal is to see how models deal with its presence. The forward feature selection is used in the case of some WoE-based models. In the case of dummy variable models it only the multicollinearity removal is considered.

The models listed in the last column are subject to hyperparameter optimization via grid search. In the case of logistic regression, L1, L2 and the elastic net penalization are tried out when multicollinearity is not removed during data preprocessing. The optimal hyperparameters of the ANN model are searched over the following grid.

Table 6: hyperparameter grid for ANN model

| **Hyperparameter** | **Combination** |
| --- | --- |
| Hidden layer sizes *(the number of integers in the bracket implies the number of layers, n integers means n layers)* | [7] [5, 5] [7, 7] [10, 5]  [10, 10]  [20, 20] [40, 40, 20]  [40, 20, 20]  [20, 40, 20] [100, 200, 100]  [200, 100, 100]  [200, 200, 100]  [200, 400, 200] |
| Activation function | Logistic Tanh Relu |
| Solver | Adam |
| Alpha | 0.001 |
| Learning rate | Constant Invscaling Adaptive |

Source: author

The optimal hyperparameters of the KNN model are searched over the following grid.

Table : hyperparameter grid for KNN model

| **Hyperparameter** | **Combination** |
| --- | --- |
| # of neighbors | From 1 to 10, by 1 |
| Weights | Uniform  Distance weighted |
| Solver | Manhattan  Euclidian |

Source: author

The optimal hyperparameters of the SVM model are searched over the following grid.

Table : hyperparameter grid for SVM model

| **Hyperparameter** | **Combination** |
| --- | --- |
| Regularization parameter | From 0 to 1, by 0.1 |
| Kernel | Linear Polynomial RBF Sigmoid |
| Polynomial degree | From 1 to 5, by 1 |

Source: author

The optimal hyperparameters of the RF model are searched over the following grid.

Table : hyperparameter grid for RF model

| **Hyperparameter** | **Combination** |
| --- | --- |
|  |  |
|  |  |
|  |  |

Source: author

### Results

#### Preprocessing

Reporting the list of selected features at each step, bin specifications, resulting WoE values, multicollinearity removal outputs etc. would be a cumbersome exercise. Since the main task of this Thesis is not to compare the results of preprocessing procedures, only several examples are shown in order to provide a general orientation of how the data is being transformed. These operations below are performed on a 70:30 train-test split with run seed set to 123.

A large number of clients that apply for a home loan already have some history with the issuing bank. The data about their behavior on their retail products, such as overdraft accounts can serve as a good indicator of how they will manage their home loan. Here, WoE grows monotonically from the lowest retail scores to the highest one. It can also be observed that a substantial part of clients did not have a behavioral retail score at the moment of applying for their home loan. Since their default rate falls in between the best-performing and the worst-performing bin, it can be expected that if their true behavioral retail scores would vary as well.

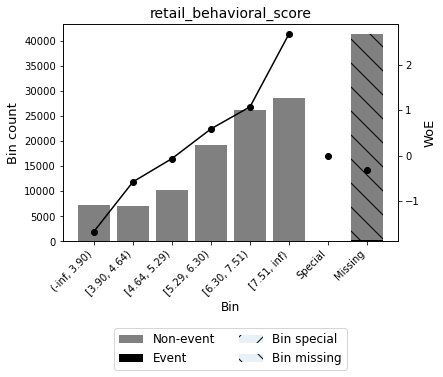


Figure : binning and WoE values of retail behavioral score, source: author

The OptBinning algorithm published by Navas-Palencia (2020) is able to group categories of a categorical variable. One such case is the marital status, where “partnered” and “single” make up distinct groups, while “divorced” and “widowed” are in a single bin. Married and partnered couples appear to be more resilient do defaults. Single individuals are not too far from the former group as well. The applicants with the remaining marital statuses have a substantially higher default rate than the previous two groups. There are nearly no missing values in the data when it comes to this feature.

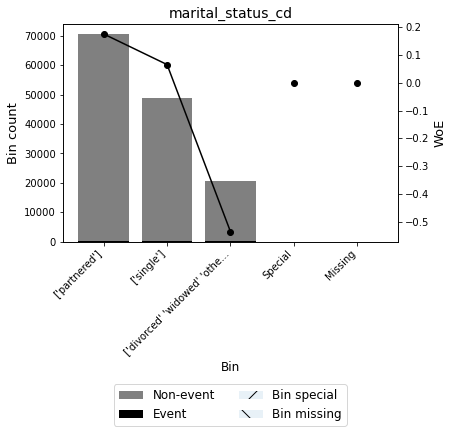


Figure : binning and WoE values of marital statuse, source: author

Although the LTV ratio has a much more direct relationship with LGD modeling, it is also reasonable to expect that clients that have more cash w.r.t. the loan amount to start with are less likely to default. What is also interesting to observe here is that the binning algorithm boiled all observations into 2 groups.

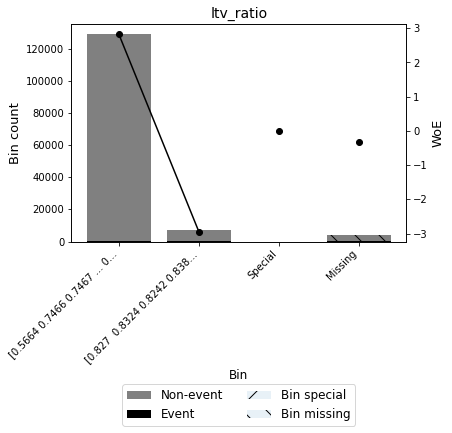


Figure : binning and WoE values of LTV ratio, source: aut hor

Upon defining the bins, a feature exclusion is performed by excluding any column that has IV lower than 0.05 or Gini lower than 0.1. This operation reduced the number of features from the initial 263 outputted from the DataGetter class to 120. These features are then passed to the multicollinearity removal procedure, where the maximum tolerated correlation between two variables is up to 0.5, as per Witzany (2017). Finally, the number of features in the shortlist is 39, they are summarized in the table below and sorted by Gini.

Table : feature shortlist obtained after data preprocessing

| **Feature** | **# of bins** | **IV** | **Gini** |
| --- | --- | --- | --- |
| behavioral\_score | 10 | 2.10 | 0.70 |
| principal\_paid\_6m\_min\_amt | 6 | 0.67 | 0.43 |
| debt\_summary\_2qs\_max\_amt | 2 | 0.71 | 0.41 |
| interest\_paid\_to\_next\_installment\_6m\_max\_ratio | 4 | 0.67 | 0.41 |
| days\_in\_deliquency\_6m\_avg\_count | 2 | 1.08 | 0.40 |
| application\_score | 10 | 0.42 | 0.35 |
| fee\_paid\_mtd\_amt | 5 | 0.46 | 0.34 |
| collateral\_required\_amt | 8 | 0.36 | 0.32 |
| principal\_paid\_to\_next\_installment\_6m\_max\_ratio | 8 | 0.39 | 0.32 |
| product\_type\_cd | 4 | 0.38 | 0.31 |
| penalty\_interest\_paid\_mtd\_amt | 2 | 0.69 | 0.30 |
| brki\_installment\_amt | 4 | 0.33 | 0.28 |
| debit\_turnover\_mtd\_amt | 4 | 0.28 | 0.28 |
| interest\_paid\_6m\_min\_amt | 7 | 0.38 | 0.27 |
| interest\_paid\_6m\_max\_amt | 9 | 0.28 | 0.26 |
| income\_to\_expense\_all\_applicants\_ratio | 9 | 0.20 | 0.24 |
| paid\_to\_limit\_ratio | 8 | 0.24 | 0.21 |
| age | 9 | 0.13 | 0.20 |
| since\_live\_acc\_opening\_mths\_count | 7 | 0.14 | 0.20 |
| main\_obj\_value\_amt | 6 | 0.19 | 0.20 |
| principal\_paid\_to\_outstanding\_6m\_max\_ratio | 8 | 0.11 | 0.18 |
| dsti\_ratio | 8 | 0.10 | 0.18 |
| od\_limit\_utilization\_amt | 2 | 0.23 | 0.18 |
| client\_capital\_to\_paid\_mtd\_ratio | 5 | 0.16 | 0.18 |
| client\_income\_amt | 6 | 0.11 | 0.17 |
| all\_applicants\_expense\_amt | 7 | 0.10 | 0.17 |
| principal\_paid\_6m\_max\_amt | 9 | 0.10 | 0.17 |
| expense\_all\_aplicants\_to\_next\_installment\_ratio | 7 | 0.09 | 0.16 |
| fixation\_to\_installments\_ratio | 6 | 0.11 | 0.16 |
| collateral\_value\_to\_limit\_ratio | 7 | 0.09 | 0.16 |
| next\_repayment\_amt | 9 | 0.08 | 0.15 |
| fixation\_period\_mths\_count | 3 | 0.11 | 0.15 |
| all\_applicants\_income\_amt | 6 | 0.08 | 0.14 |
| principal\_paid\_to\_outstanding\_3m\_avg\_ratio | 9 | 0.07 | 0.13 |
| refinancing\_flg | 2 | 0.15 | 0.12 |
| marital\_status\_cd | 3 | 0.07 | 0.12 |
| debt\_summary\_mtd\_amt | 1 | 0.06 | 0.12 |
| main\_object\_as\_coll\_flg\_2 | 2 | 0.05 | 0.11 |
| live\_saving\_account\_prev\_mth\_flg | 2 | 0.09 | 0.11 |

Source: author

#### Models

==============================================================================

Dep. Variable: default\_event\_flg No. Observations: 4008

Model: Logit Df Residuals: 3993

Method: MLE Df Model: 14

Date: Thu, 25 Apr 2024 Pseudo R-squ.: 0.1649

Time: 22:33:27 Log-Likelihood: -1262.7

converged: True LL-Null: -1512.0

Covariance Type: nonrobust LLR p-value: 1.795e-97

==============================================================================

coef std err z P>|z| [0.025 0.975]

------------------------------------------------------------------------------

x1 -0.3127 0.168 -1.865 0.062 -0.641 0.016

x2 -0.3560 0.083 -4.276 0.000 -0.519 -0.193

x3 -0.0441 0.086 -0.514 0.607 -0.212 0.124

x4 -0.8367 0.130 -6.433 0.000 -1.092 -0.582

x5 -0.2962 0.073 -4.034 0.000 -0.440 -0.152

x6 -0.6069 0.218 -2.784 0.005 -1.034 -0.180

x7 -0.4669 0.212 -2.202 0.028 -0.882 -0.051

x8 -0.5696 0.107 -5.313 0.000 -0.780 -0.359

x9 -10.3361 0.487 -21.229 0.000 -11.290 -9.382

x10 -0.4758 0.126 -3.784 0.000 -0.722 -0.229

x11 -0.7640 0.213 -3.588 0.000 -1.181 -0.347

x12 -0.4472 0.163 -2.748 0.006 -0.766 -0.128

x13 -0.2654 0.086 -3.099 0.002 -0.433 -0.098

x14 -0.1878 0.074 -2.526 0.012 -0.333 -0.042

x15 -0.5167 0.048 -10.773 0.000 -0.611 -0.423

==============================================================================

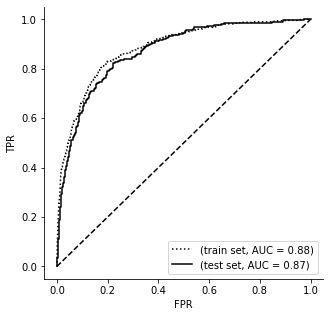


Figure 7.

### Comparison

# Final Remarks

# References

Altman, E. (1968). FINANCIAL RATIOS, DISCRIMINANT ANALYSIS AND THE PREDICTION OF CORPORATE BANKRUPTCY. *The Journal of Finance*. *23*(4), 589–609. doi: 10.1111/j.1540-6261.1968.tb00843.x

Barboza, F., Kimura, H. & Altman, E. (2017). Machine learning models and bankruptcy prediction. *Expert Systems with Applications*, *83*, 405–417. doi: 10.1016/j.eswa.2017.04.006

European Banking Authority. (2017). *Guidelines: on PD estimation, LGD estimation and the treatment of defaulted exposures* (EBA/GL/2017/16).

European Banking Authority (2021). *EBA discussion paper on machine learning for IRB models* (EBA/DP/2021/04).

European Banking Authority. (2023). *Machine learning for IRB models: follow-up report from the consultation on the discussion paper on machine learning for IRB models* (EBA/REP/2023/28).

European Parliament. (2023). *EU AI Act: first regulation on artificial intelligence*. Retrieved April 1, 2024, from https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence

Engelmann, B., Hayden, E. & Tasche, D. (2003). Measuring the Discriminative Power of Rating Systems. *Deutche Bundesbank, Frankfurt a. M., Discussion Paper Series 2, 2003(01)*. Working paper.

Ernst & Young. (2021). *2EK614 Applied Operational Research in Business Consulting: Credit risk – Predictive Modelling*. Prague University of Economics and Business.

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

Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of Research. *European Journal of Operational Research*, *247*(1), 124–136. doi: 10.1016/j.ejor.2015.05.030

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

*Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) 2016*. (EU)

Shi, S., Tse, R., Luo, W., D’Addona, S. & Pau, G. (2022). Machine learning-driven credit risk: a systemic review. *Neural Computing and Applications*, *34*, 14327–14339. doi: 10.1007/s00521-022-07472-2

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

# 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