Digital Finance
https://doi.org/10.1007/s42521-023-00097-7

RESEARCH

Fast approximation methods for credit portfolio risk calculations

Kevin Jakob[1] · Johannes Churt[2] · Matthias Fischer[3] · Kim Nolte[2] · Yarema Okhrin[1] · Dirk Sondermann[2] · Stefan Wilke[2] · Thomas Worbs[2]

Received: 6 May 2023 / Accepted: 26 September 2023

Abstract
Credit risk is one of the main risks financial institutions are exposed to. Within the last two decades, simulation-based credit portfolio models became extremely popular and replaced closed-form analytical ones as computers became more powerful. However, especially for non-homogenous and non-granular portfolios, a full simulation of a credit portfolio model is still time consuming, which can be disadvantageous within some use cases like credit pricing or within stress testing situations where results must be available very quickly. For this purpose, we investigate if methods based on artificial intelligence (AI) can be helpful to approximate a credit portfolio model. We compare the performance of AI-based methods within three different use cases with suitable non AI-based regression methods. As a result, we see that AI-based methods can generally capture portfolio characteristics and speed-up calculations but - depending on the specific use case and the availability of training data - they are not necessarily always the best choice. Particularly, considering the time and costs for collecting data and training of the complex algorithms, non-AI-based methods can be as good as or even better than AI-based ones, while requiring less computational effort.

Keywords Credit risk · AI · Credit portfolio model · Approximation

JEL Classification C63 · G32

1 Introduction

In many financial institutions, credit portfolio models are used to estimate the loss distribution of the credit portfolio and to calculate corresponding risk figures such as value at risk or expected shortfall (McNeil et al. 2015). The tail of the loss distribution is mainly determined by the correlation between default events or rating changes of individual obligors if they represent a significant share of the portfolio

Extended author information available on the last page of the article

(concentration risk). Capturing these correlations precisely results in quite complex mathematical models. There are some closed-form models, e.g. the Vasicek model (Vasicek 2002) or the CreditRisk+ approach (Credit Swiss 1997), which, however, require strong and to some extent unrealistic assumptions on portfolio structure, stochastic distributions, and correlations. For example, the Vasicek model assumes a portfolio of infinite granularity, i.e. without any concentration risk on counterparty level, whereas the CreditRisk+ model in its basic form assumes Poisson distributed defaults and independent economic sectors.

Thus, these models do not suffice for real-world application within the internal capital adequacy assessment process (ICAAP) of financial institutions. Therefore, most credit risk portfolio models rely on Monte Carlo techniques to simulate defaults, rating migrations, or changes of asset values (e.g. Gupton et al. 1997). Nowadays, computer performance improves more and more but the estimation of the loss distribution and corresponding risk measures on a portfolio level is still time-consuming. Calculations that take hours may be acceptable for monthly checks of risk bearing capacity, but neither what-if-scenarios for risk management or calibration purposes nor pricing considerations can be practically supported by Monte-Carlo-simulation-based models if results are needed within minutes. To address this issue, several variance reduction techniques (like importance sampling, see e.g. Glasserman 2003)) are available or one can use specific approximation techniques for the portfolio loss distribution as the ones described by Grundke and Moosbrucker (2008). However, in this paper, we want to investigate different methods to achieve a fast and accurate approximation of the full credit portfolio Monte Carlo simulation. In particular, we compare how Artificial Intelligence (AI) techniques perform compared to suitable non AI-based regression methods.

AI in risk management has become a field of active research. We refer to Leo et al. (2019) or Aziz and Dowling (2019) for an overview of AI methods and applications in different areas of risk management as well as credit risk. Here, we make use of the established approach of training an AI model to approximate a non-linear function (i.e. the loss distribution depending on portfolio and correlation parameters) that is computationally expensive to calculate otherwise (see Liu et al. (2019)). To the best of our knowledge, this approach has not been applied to credit portfolio models yet.

In practice, in the context of loan pricing (e.g. see Chun and Lejeune (2020) or Duffie (2003)) or scenario analysis, in most cases it is not necessary to obtain the same level of precision as in case of the monthly ICAAP reporting. Therefore, approximations of the full credit portfolio model are used, e.g. to obtain a risk contribution for an additional loan in the portfolio. These approximations are often quite ad-hoc and do not consider much of the complex interactions (i.e. due to correlations) between the loans of the portfolio.

We evaluate the performance of different approximation methods on three use cases, namely to predict

- the total value at risk for a credit portfolio,
- the risk contribution of an individual (additional) counterparty, and
- the portfolio value at risk given a change in model (i.e. correlation) parameters.

The second case can be used to assess the effect of additional loans in the portfolio (e.g. during initial business contact), while the first and the third can be used in the context of calibration and stress testing (i.e. if economic circumstances changes significantly).

The paper is organized as follows: In Sect. 2, we give a brief overview of the credit portfolio models in focus. Section 3 briefly presents the AI and non AI-based methods used. In Sect. 4, we describe our simulation study in detail, followed by a summary and outlook in Sect. 5.

2 Overview of credit portfolio model

The credit portfolio model we use within our study is a simulative framework of CreditMetrics type, i.e. the link function between sector variables and counterparties PD (see Sect. 3) is of CreditMetrics type (see Gupton et al. (1997)). In this section, we introduce the model setup and distributional assumptions, followed by a short description of the implementation.

Without loss of generality, we assume that our loan portfolio is already aggregated on counterparty level. Therefore the portfolio contains $M \in \mathbb{N}_{>0}$ counterparties CP_i with $i = 1, ..., M$. The maximum exposure to each counterparty which can be lost in case of a default is described by $\mathrm{EXP}_i > 0$. In case of a default of counterparty i the loss is given by $L_i := \mathrm{EXP}_i \cdot \mathrm{CCF}_i \cdot \mathrm{LGD}_i$, where CCF_i describes the so-called credit conversion factor and LGD_i the loss given default.

To keep the model setup simple and to focus on the different approximation methods, we assume that $\mathrm{CCF}_i = 1$ and $\mathrm{LGD}_i = 1$ for each counterparty. Of course, within real world applications a $\mathrm{CCF} < 1$ is necessary for contingent liabilities such as credit commitments or credit avals to account for the probability that these positions are actually used. Furthermore, the LGD describes the share of the exposure that will be lost, i.e., which is not covered by guarantees, other types of collateral or which cannot be recovered during the liquidation process. However, since both CCF and LGD are not modelled stochastically within our framework, it is not a crucial restriction to assume that both are constant and equal to 1. Furthermore, CCF and LGD affect the resulting loss only in a linear way, whereas we want to focus on the approximation of non-linear relationships (i.e. between counterparties' creditworthiness and the risk figures). Therefore, we will use the terms EXP_i and $\mathrm{EAD}_i := \mathrm{EXP}_i \cdot \mathrm{CCF}_i$, which denotes the exposure at default interchangeably.

The probability of default for counterparty i is denoted by PD_i. We assume this as an exogenous variable estimated via an internal or external rating system. I.e., the estimation of PD_i is not part of the credit portfolio model framework or our approximation methods.

To introduce a dependence structure between counterparties, we assume that each of them belongs to one of $K \in \mathbb{N}_{>0}$ sectors. The model can be easily extended to a model with multiple sectors per counterparty including a vector of asset correlations and a sector correlation matrix. However, to concentrate on the AI approximation methods, we restrict the model framework to one sector per counterparty, which is still very relevant for real world applications. Typically, a

sector represents an industry and/or country of risk the counterparty (primarily) belongs to, but it can also be used for other characteristics influencing the creditworthiness of the counterparty. The current state of the kth sector is denoted by S_k with $k = 1, ..., K$. S_k is a stochastic variable distributed according to the standard normal distribution, i.e. $\mathcal{N}(0, 1)$. Given a realization s_k of S_k, the creditworthiness index (also named ability to pay variable or asset value) is modeled by

$$B_i := R_{k(i)} s_k + \sqrt{1 - R_{k(i)}^2}\, \epsilon_i, \tag{1}$$

with model variable $R_{k(i)}$ determining the correlation of counterparty i's creditworthiness with the realization s_k of sector $k(i)$ it belongs to and $\epsilon_i \sim \mathcal{N}(0, 1)$ for all i.

Since $R_{k(i)}$ affects the correlation structure between different counterparties and therefore significantly influences the estimation of the portfolio loss distribution, its estimation is an important issue. However, this is not within the focus of this study. For more information on this topic, we refer to Kalkbrener and Onwunta (2010), Düllmann et al. (2010), Dorfleitner et al. (2012), Gordy and Heitfield (2010) or Pfeuffer et al. (2019). A default occurs if the creditworthiness B_i falls below a certain default threshold determined by PD_i. To indicate a default event, we use the default variable

$$D_i := \mathbb{I}_{B_i \leq \Phi^{-1}(\mathrm{PD}_i)}, \tag{2}$$

where $\mathbb{I}$ denotes the indicator function and Φ^{-1} the quantile function of the standard normal distribution.

Using Eqs. (1) and (2), the conditional probability of default PD_i^S given a realization s_k reads

$$\mathrm{PD}_i^S(s_k; \mathrm{PD}_i, R_{k(i)}) := \Phi\left(\frac{\Phi^{-1}(\mathrm{PD}_i) - R_{k(i)} s_k}{\sqrt{1 - R_{k(i)}^2}} \right). \tag{3}$$

Typically, the multivariate sector distribution of $\boldsymbol{S} = \left(S_1, ..., S_K\right)$ is modeled by a multivariate normal distribution $\boldsymbol{\mathcal{N}}(\boldsymbol{\mu}, \Sigma)$ with mean vector $\boldsymbol{\mu} = 0$ and correlation matrix $\boldsymbol{\Sigma}$. Please note that to ensure $S_k \sim \mathcal{N}(0, 1)$, the variances of all sectors must be equal to one (i.e. $\boldsymbol{\Sigma}$ is truly a correlation, not only the covariance matrix). But, using Sklar's theorem (see Sklar (1959)) from copula theory, it is also possible to create a new multivariate distribution function $\mathcal{S}$ for $\boldsymbol{S}$ with an arbitrary copula function (determining the dependence structure between different sectors) and using $\mathcal{N}(0, 1)$ for the marginal distributions.

Within our simulation study, we use the implementation of the model described above within the GCPM R-package. The package includes a simulative framework with the link function (3) of the CreditMetrics type. The simulation process described in Jakob and Fischer (2016) contains the following steps for $N > 0$ simulations:

Algorithm 1 Simulation Algorithm

for $n = 1, ..., N$ (simulation loop)
  draw sector realizations $\boldsymbol{s}^n = (s_1^n, ..., s_K^n) \sim \mathcal{S}$
  for $i = 1, ..., M$ (counterparty loop)
    calculate conditional PD: $\mathrm{PD}_i^S := \Phi\left(\frac{\Phi^{-1}(\mathrm{PD}_i) - R_{k(i)} s_k}{\sqrt{1 - R_{k(i)}^2}}\right)$
    default drawings: $D_i \sim \mathrm{Bern}\left(\mathrm{PD}_i^S\right)$
    determine counterparty loss: $L_i^n = \mathrm{EXP}_i D_i$
  determine portfolio loss: $L^n = \sum_{i=1}^M L_i^n$

Given the N different realizations $L^n, n = 1, ..., N$, one can easily estimate the distribution or density function of the portfolio loss by means of the respective empirical counterparts. Given the loss distribution, portfolio risk figures such as value at risk VaR_α, which is defined as the α-Quantile of the portfolio loss, or expected shortfall

$$\mathrm{ES}_\alpha := \frac{\sum_{n=1}^N \mathbb{I}_{L^n \geq \mathrm{VaR}_\alpha} L^n}{\sum_{n=1}^N \mathbb{I}_{L^n \geq \mathrm{VaR}_\alpha}},$$

which is defined as the conditional mean given a portfolio loss greater or equal to the value at risk on level $\alpha \in (0, 1)$, can be determined.

To analyze the portfolio risk figures in more detail, it is also possible to calculate so-called risk contributions on a counterparty level. The counterparty risk contributions to expected shortfall are estimated based on the mean loss of the counterparty within expected-shortfall-relevant portfolio simulations and denoted by $\mathrm{RC}_i^{\mathrm{ES}_\alpha}$. Within the GCPM package, it holds true that $\sum_{i=1}^M \mathrm{RC}_i^{\mathrm{ES}_\alpha} = \mathrm{ES}_\alpha$. For more detailed information, please refer to Jakob and Fischer (2016) or the package documentation.

Using the described credit portfolio model, we can calculate risk figures like VaR on portfolio and on counterparty level to create our training and test data. Within the next section, we introduce the AI methods we are going to use to approximate and speed up the calculation of the credit portfolio model.

3 AI-based approximation methods

In this section, we give a brief overview of the AI methods used in this paper.

The birth of the artificial intelligence discussion dates already back to Alan Turing's seminal work on "Computing Machinery and Intelligence" in Turing and Haugeland (1950). Roughly speaking, AI describes a broad field of computer science where the focus is on a machine's capability to produce rational behavior from external inputs. The goal of AI is to create systems that can perform tasks that would otherwise require human intelligence. The strongly growing popularity of AI mainly results from the recent advancements in the sub-field of machine learning (ML).

Machine learning algorithms can be subdivided into four classes: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Due to the nature of our problem to predict features (real number output values) from a given set of labels (real number input values), our study explores various supervised regression machine learning methods and their suitability for our simulation. In supervised regression learning, an algorithm attempts to derive a function from a given training data set with label-feature pairs. The function is optimized to accurately predict new, unknown feature (output) values from new label (input) values.

3.1 Artificial neural networks

Recently, artificial neural networks (ANNs) have been very successful at machine learning tasks like those mentioned above. ANNs are based on a collection of connected nodes, the so-called neurons. In the common variant of a sequential ANN, each connection, like the synapses in a biological brain, can transmit a signal to other neurons. After receiving a signal, the neuron processes it and forwards signals to connected neurons. The signal at a connection or edge is a real number, and the output of each neuron is computed typically by some linear or non-linear function (activation function) of the sum of its inputs.

In addition, neurons and edges typically have a weight that varies as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may also have a threshold such that a signal is sent only if the aggregate signal exceeds that threshold. In sequential ANNs, neurons are grouped into so-called layers. Different layers may perform different transformations on their inputs. In the chosen network, signals travel from the first layer, the so-called input layer, to the last layer, the output layer. All other layers in between are termed hidden layers.

For a specified loss function, optimization algorithms (optimizers) are applied to determine the unknown parameters of the ANN for a suitable training data set. For more details on ANN's, see for instance Huang et al. (2020) or Dixon et al. (2020).

3.2 Alternative AI methods

Besides ANNs, many other machine learning methods for supervised regression learning have been successfully applied. Therefore, we also investigate the suitability of other AI-based algorithms in this study. In particular, we test various tree-based models like the rule-based model CUBIST method, Support Vector Machines, XGBoost or Random Forest method or piece-wise regression with Multivariate Adaptive Regression Splines and the K-Nearest Neighbors Algorithm for regression problems.

3.2.1 CUBIST method

CUBIST is a rule-based multivariate regression model founded mainly on the work of Ross Quinlan (see Kuhn and Quinlan 2022). The CUBIST models a decision tree,

where the nodes are linear regressions using as many labels as the level of the node. After setup the tree is simplified again by removing branches without effect on the error or by combining similar branches. The final ruleset of the model consists of the linear regressions assigned to the leaves of the tree.

For a "boost-like" capability, the CUBIST generates iterative decision trees during training, called "committees". Each subsequent tree has the additional goal to adjust over- or under-predictions. The final prediction is formed by averaging across all committees.

3.2.2 Support Vector Regression (SVR) method

The difference between "plain" regression models like linear regression and SVR is the optimization target. While plain models minimize the error rate itself, SVR aims to find a best fitting hyperplane in the typically multidimensional label/feature space allowing a given absolute error threshold band of ε around this hyperplane. The optimization target of the SVR is not to minimize the error itself but to minimize the L^2-Norm of the linear coefficients together with the restriction that the absolute error is located within the ε band. This may result in better-adopting models that are more robust compared to the plain ones. Another big advantage of SVR is its support of non-linear relationships between input and output variables within data sets using the so-called "kernel trick". Kernels are functional transformations of the label/feature space into a higher dimensional space to make them separable by linear hyperplanes despite their non-linear dependencies. An introduction to SVR with further details can be found here (Premanand 2021).

3.2.3 Random Forest™ method

The Random Forest™ regression method has its roots in Classification And Regression Trees (CARTs). Random Forest™ sets up an ensemble of B CARTs when trained but, in contrast to XGBoost described in the next section, these trees are independent. Each of the B trees is trained from a separate training data set D_b (for $b \in \{1, \ldots, B\}$) that is randomly sampled from the whole training set D with replacement meaning D_b may contain the same sample d_i of the original training set D multiple times. This sampling method is called Bootstrapping. Within a typical implementation, the predictions of all B trees are aggregated simply by calculating the mean of the predictions of all B trees after the training. This aggregation step together with the aforementioned Bootstrapping, is called Bagging. In Random Forest™ it has several advantages compared to plain CARTs. It especially avoids overfitting, the model is very robust, and it allows highly parallel computing through the independent trees (Sruthi 2022).

The first algorithm for a Random Forest™ was designed in 1995 by Tin Kam Ho (see Ho 1998). Leo Breiman and Adele Cutler (see Breiman 2001) extended the algorithm with Breiman's "Bagging" idea later in 2001.

3.2.4 Xtreme Gradient Boosted Trees (XGBoost) method

XGBoost is a relatively young machine learning method developed by Tianqi Chen and Carlos Guestrin resulting from a research project of the University of Washington (Chen and Guestrin 2016).

The XGBoost algorithm combines three machine learning techniques: CART, the ensemble method of Boosting, and the use of a Taylor expansion of the loss function up to the second derivative considering the average gradient of the loss as the criterion to be minimized with each subsequent boosting step. CARTs are binary decision trees taking a decision on each leaf based on the value of a specific label (less or equal or greater than a learned decision boundary) to determine the path to the next leaf (see Breiman 1984). Boosting combines multiple sequentially arranged CARTs in a manner that with each new tree the feature prediction of the previous tree is improved by an additive component. Starting with an initial regression function $F_0(x)$ simply delivering the mean of a feature of the training data set samples, a sequence of CARTs is generated using the residual errors of the predecessor as additional label input to produce an additive correction as the output feature refining the overall prediction step by step. When it comes to the optimization of the CART leaf splits, XGBoost uses the mean gradient of the loss improvements for a new refining CART. For more details please refer to Bhaskar Sundaram (2022) and Chen and Guestrin (2016).

4 Simulation study

After presenting the credit portfolio model in Sect. 2, which provides us the objective values (i.e., VaR) we want to approximate and the several AI-based methods describe in Sect. 3 we now compare them within a simulation study to non AI-based methods. In the following sections, we describe the overall simulation framework (i.e. the data generation process) and the quality measures we apply. Afterwards, we compare the approximation performance of the different methods in three use cases mentioned in Sect. 1.

The simulation framework is implemented in R using several packages like Keras, caret and Tensorflow for the AI based methods and the GCPM, MLmetrics and copula packages to obtain training and test data to evaluate the approximation performance. The training of the AI-based methods was performed in an environment with 32 CPU cores and 4 GPUs (NVIDIA Tesla V100), while the generation of training data, which does not benefit from GPU support, was performed in an environment with 72 CPU cores and no GPUs.

4.1 Data generation and simulation preparation

The data sets for all use cases consist of synthetic credit portfolios. Due to the very time-consuming CPU-intensive process of data generation and training we simply

split our data set into training and test data by 70 to 30 instead of using a classical cross validation.

Depending on the use case, each credit portfolio consists of up to 20,000 counterparties. As described in Sect. 2, each counterparty i has the characteristics EAD_i, PD_i (which is discretized into one of 21 rating classes RG_i), a business sector $k(i) \in \{1, 2, 3\}$ determining the correlation parameter $R_{k(i)}$, and value at risk on level $\alpha = 99.9\%$ denoted by $\mathrm{VaR}_{\alpha,i}$. The value at risk of a counterparty is calculated by the credit portfolio model (see Sect. 2) as the counterparty's risk contribution $\mathrm{RC}_i^{\mathrm{ES}_\tau}$ to the portfolio's expected shortfall ES_τ on confidence level τ, such that on portfolio level $\mathrm{ES}_\tau = \mathrm{VaR}_\alpha$ for confidence level $\alpha = 99.9\%$. All characteristics are summarized in Table 1.

In addition to the VaR, we also use a VaR weight defined by

$$\mathrm{VaRW}_i := \frac{\mathrm{VaR}_i}{\mathrm{EAD}_i}, \tag{4}$$

which, in case it is not constant across different counterparties, characterizes the non-linear relationship between the EAD and the VaR and which varies much more slowly with the EAD.

The business sector $k(i)$ of counterparty i is independent of its EAD_i and PD_i. The sector correlation parameter R strongly influences the VaR of the portfolio. The values typically observed vary depending on the specific sector, the portfolio and the data used for estimation (see Düllmann et al. (2010)). For simplification, we use only $K = 3$ business sectors for the portfolio, representing sectors with high ($R_1 = 0.30$), medium ($R_2 = 0.15$) and low correlation ($R_3 = 0.05$) parameter. This covers the range of correlation parameters usually observed (e.g. see Düllmann et al. (2010), Lee et al. (2011), Hahnenstein (2004), Geidosch (2014) or Akhavein et al. (2005)). We randomly assign one of the sectors to each counterparty. The dependency between business sectors is modeled via a t-copula with 3 degrees of freedom and fixed correlation parameter of 0.5. A short explanation of the copula concept and the t-copula is given in Appendix A.

In practice, credit risk applications often use discrete rating classes instead of a continuous PD. Therefore, the PD_i is discretized into 21 rating classes $\mathrm{RG}_i \in \{1, \ldots, 21\}$. To achieve this, the rating classes are exponentially spaced between the extreme values 0 and 1 for the PD. Each counterparty is assigned the rating class matching its PD.

Table 1 Characteristics of a counterparty

Characteristic	Source
Exposure at default EAD_i	Drawn from distribution
Probability of default PD_i	Drawn from distribution
Value at risk VaR_i	Credit Portfolio Model (Sect. 2)
Rating class RG_i	Discretization of PD_i
Business sector affiliation $k(i)$	Random assignment

4.1.1 Drawing EAD and PD

To create realistic training and test data, we use the following distributions and parameters. Typically, the creditworthiness of counterparties is not symmetrically distributed but right skewed such that the share of counterparties with higher PDs is lower. Therefore, we use a gamma distribution $\Gamma\left(1, \frac{1}{100}\right)$ with parameters such that $\mu = 0.01$ and $\sigma = \mu$ to simulate the PDs across the portfolio. Usually, the EAD distribution is also right-skewed. However, to introduce more concentration risk, we use a log-normal instead of a gamma distribution with $\mu = 10^6$ and $\sigma = 10^{14}$. Furthermore, from empirical observations, we know that the EAD and PD of a counterparty are not independent (Jacobs 2010). To take this into account, we use a t-copula with dispersion parameter -0.5 and three degrees of freedom to allow for some tail dependencies.

4.1.2 Intensifying Concentration Risk

Real-life credit portfolios tend to be concentrated with respect to EAD, and the resulting concentration risk can strongly influence the portfolio VaR. The chosen log-normal distribution of EAD already creates a certain level of concentration risk. However, depending on the use case, we enlarge the EAD of the 10 counterparties with the highest exposure by a random factor in the range between 1 and 10 to intensify the concentration risk and to challenge our AI and non AI-based approximation methods further.

4.1.3 Reducing Intrinsic Error of the Monte Carlo simulation

The VaR_i of a counterparty is calculated with the help of a Monte Carlo simulation. Thus, the stability of the VaR_i figures, i.e. the intrinsic simulation error, represented by the RMSPE of VaR_i over all simulations of the data-generating process is limited by the number of simulation loops within the Monte Carlo simulation. Especially within the use case of Sect. 4.3 this was a crucial problem. For each portfolio setup we determined the maximum number of loops possible within the memory limit of our system.

If the resulting instability was too high to evaluate the performance of the approximation methods because the approximation error could not be separated from the simulation error, we repeated the Monte Carlo simulation multiple times and averaged the result over the runs. Since we need this kind of technique only in use case 2 to reduce the simulation error down to 0.28% within an acceptable amount of time, we decided to use this simple and convenient way instead of other more complex methods like importance sampling to reduce the simulation error.

4.1.4 Quality measures

To quantify the error of our approximation, we used the root mean square percentage error $\text{RMSPE} := \sqrt{\frac{1}{N}\left(\frac{x-y}{y}\right)^2}$ as relative error measure, where y denotes the

true VaR_i from the data generating process and x denotes the model prediction. Since the approximation results in our use cases are some variant of the VaR, the error is influenced by the quality of the data generating process (i.e., the Monte Carlo simulation). Therefore, we will compare the RMSPE of our approximations to the intrinsic error of the Monte Carlo simulation for the use cases.

4.2 Use Case 1: predicting the total VaR of a credit portfolio

The objective of the first use case is to predict the overall portfolio VaR as a function of the portfolio composition, i.e. the individual counterparty parameters. The systemic risk parameters (sector correlations) were held fixed in this use case. Please note, that predicting the VaR on portfolio level does not imply a prediction of VaR contributions on counterparty level. Even if one could in principle calculate the VaR difference of a base portfolio and the same portfolio with a given counterparty added, the inevitable uncertainties of the approximations would render this difference meaningless.

4.2.1 Approximation using an ANN

4.2.1.1 Drawing portfolios The portfolios for this use case are drawn as described in Sect. 4.1, applying the additional step of enlarging concentration risk to provoke more non-linear effects i.e. a disproportionate increase of the VaR due to an increasing EAD. Since the simulation error is much smaller compared to the use case in Sect. 4.3 we did not apply simulation averaging.

4.2.1.2 Data preparation The portfolio contains parameters PD, EAD, and business sector for each counterparty. Therefore, a portfolio with M counterparties corresponds to $3M$ variables. Regarding the amount of available training data, this number of variables is too high to train the ANN and to prevent over-fitting. Since the risk figures of a credit portfolio do not depend on the order of counterparties, we apply the following binning approach, which significantly reduces the test data and computing power requirements.

- The EAD_i of a counterparty is normalized by replacing it with the portfolio share $\text{EAD}_i / \sum_j \text{EAD}_j$.
- These normalized values are mapped to 21 buckets using equidistant cuts in the range from 0 to the maximum value across all portfolios.
- The PD_i of a counterparty is discretized by its assignment to a rating class RG_i.
- Combining an EAD bucket $e \in \{1, \ldots, 21\}$, a rating class $\text{RG} \in \{1, \ldots, 21\}$ and a business sector $k \in \{1, 2, 3\}$ results in a representation $b = (e, \text{RG}, k)$ for each counterparty with a total of $21 \cdot 21 \cdot 3$ buckets.
- For each bucket b, the number of counterparties C_b and the mean EAD ratio E_b are calculated.

- The number of buckets is reduced by merging sparsely filled buckets (e, RG, k) with the next densely filled bucket (e', RG, k) or, if there is no such densely filled bucket, with (e, RG', k). Here, a bucket b is considered sparsely filled if the average of C_b across all portfolios does not exceed 2. The figures C_b und E_b are recalculated for the merged buckets. This merging step reduces the number of bins from $21 \cdot 21 \cdot 3$ to 105.

As a result of binning, we end up with a total of 210 variables C_b and E_b. We use the transformed quantities $D_b = C_b \cdot E_b$ and E_b as input labels for the ANN. Since the overall VaR of a portfolio increases linearly if all EAD_j are scaled by a certain factor (this is just a scaling invariance with respect to the currency unit), we use the invariant ratio $V = \mathrm{VaR} / \sum_j \mathrm{EAD}_j$ as output label and thereby reduce the approximation problem by an unnecessary degree of freedom.

By construction, the input labels are always located on the hyperplane $\sum_b D_b = 1$. Using only training data with this property could result in a model that is not robust to input data that deviate from the hyperplane, e.g. due to rounding errors. Therefore, we improve the robustness of the resulting model by extending the training data by observations in an environment of the hyperplane. These additional observations are derived from the original training data by using the scaling property described above, i.e., by scaling a VaR and the corresponding EAD_j by the same factor f. For each original observation, we choose 20 random factors $f = e^{\lambda}$, where λ is drawn from a uniform distribution in the interval $\left[-\frac{1}{5}, \frac{1}{5}\right]$, and add new observations with $\mathrm{VaR}' = f \cdot \mathrm{VaR}$ and $\mathrm{EAD}'_j = f \cdot \mathrm{EAD}_j$ to the training data set. At the level of the input and output labels defined above, this means $V' = f \cdot V$, $D'_b = f \cdot D_b$ and $E'_b = f \cdot E_b$.

Table 2 Setup of the ANN and credit portfolios

Parameter	Value
Number of layers	5
Neurons per layer	210 - 210 - 210 - 70 - 1
Activation function	ReLu - ReLu - ReLu - ReLu - Linear
Regularizer	L1(0.02) - L1(0.04) - L1(0.06) - L1(0.08) - none
Initializer	He normal
Optimizer	Adam
Learning rate	$12 \cdot 10^{-5}$
Learning rate decay	$1 \cdot 10^{-5}$
Learning weight	1
Maximum number of epochs	800
Batch size	720
Input labels	Transformed EAD and number of counterparties per bin
Output label	Transformed VaR on portfolio level
Number of portfolios	95000
Number of counterparties per portfolio	1000

Based on our training and test data we evaluated several ANNs with different settings. We varied the number of layers and neurons, activation functions, batch size and other parameters. Finally, with respect to the approximation performance we choose the setup described in Table 2 with the transformations on EAD, number of counterparties and VaR as described above.

4.2.2 Alternative methods and discussion of results

As already mentioned in Sect. 3, besides the ANN method we also test the suitability of several other AI methods on the prediction of the total VaR of a credit portfolio. Like in case of the ANN, the output label of all other methods is the transformed value of the VaR on portfolio level (see 4.2.1). Due to limited computing power, we only used as a subset of 1000 counterparties of the whole data set for alternative AI methods.

Given our test data the ANN results in a RMSPE of 3.2%. The error decreased very quickly with respect to the epochs. The ANN is followed by the CUBIST method with 100 committees as the best of our alternative AI-methods with a RMSPE of 3.9% and the two SVRs with radial and linear kernel function with 4.2% and 4.4% respectively. The computational cost of the SVR training was extremely high, orders of magnitude above all other models mentioned, and at the limit what we could afford with our environment. When applying the linear kernel we had to cut down our data set to 15,000 samples to get any result at all in reasonable computing time. Additionally, as a simple benchmark to the ANN we also used a linear regression $V = \sum_b (\delta_b D_b + \gamma_b E_b)$ with parameters δ_b, γ_b and the same buckets b and input features D_b, E_b used for the ANN. This results in a RMSPE of 4.4%, which is only slightly worse than the 3.2% of the ANN and on the same level as

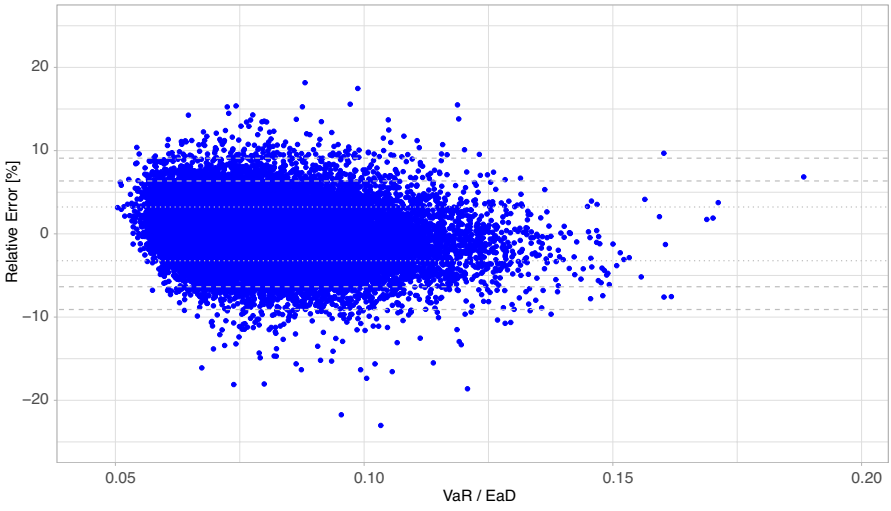

Fig. 1 Relative error for the ANN prediction, dotted line is the RMSPE, dashed lines are 95%- and 99% -confidence intervals, data points are the different portfolios

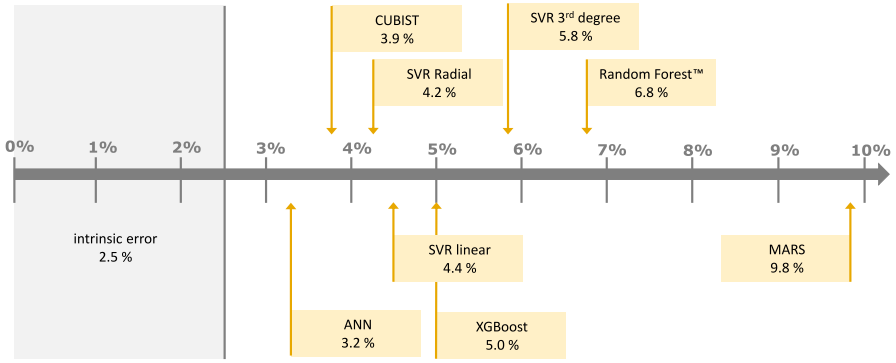

Fig. 2 Overview of approximation results from AI-based and non-AI based models

the linear SVR. Taking into account the significantly lower computational cost of the linear regression this should be preferred over the SVRs. The great number of input labels prevents the application of non-linear regression models. Even with a quadratic model, the number of regression parameters would have the same order of magnitude as the number of observations in the training data set. The XGBoost method leads to RMSPE of 5.0% after a series of hyperparameter optimization cycles. Regarding computing effort for training cycles XGBoost performs extremely well, far ahead of SVRs and Random Forest™. We could use the full 63,000-records training set without any problem, completing the training in a few minutes. With the Random Forest™ we achieve a RMSPE result of 6.79% after a rudimentary hyperparameter optimization, which is roughly 1.7% worse than XGBoost. The computing cost of the training is not in the high range of SVRs, but far higher than CUBIST and XGBoost. Therefore we restrict the data set from 63,000 to only 31,000. We also evaluate some other methods like CART, MARS, or K-nearest neighbors with worse results. All results are summarized in Figs. 1 and 2.

Besides the RMSPE we also evaluated the maximum single outlier for all methods. Within the ANN experiment, the worst single outlier is well within the boundary of 25% regarding the relative error between output feature and prediction. For the best-performing other method, CUBIST, we find an 47.5% outlier. The only method performing in a similar outlier range as the ANN is the SVR with a linear kernel with a maximum relative error of 22.5%.

4.3 Use case 2: predicting VaR on counterparty level

Within the loan origination process, the contribution of a new counterparty or a new loan to the overall portfolio VaR is an important information. This is the case because the portfolio VaR is usually subject to limits set within the ICAAP. Furthermore, in the context of risk-based pricing, the marginal risk contribution is relevant for credit pricing.

Using a classical Monte Carlo simulation, the calculation of marginal risk contributions would be very time and resource consuming. Thus, banks could benefit

from a quick and more efficient way of calculating VaR contributions for new loans. Therefore, the aim of this use case is to approximate the counterparty specific VaR contribution to an otherwise fixed portfolio depending on idiosyncratic parameters like EAD, PD, or sector.

This is performed using an ANN Sect. 4.3.1 and a non AI-based approximation method Sect. 4.3.2.

4.3.1 Approximation using an ANN

4.3.1.1 Creation of training and test data Since we are interested in the contribution of an individual counterparty to the VaR of a given portfolio, we could in principle create a fixed portfolio of $M-1$ counterparties and then add a randomly drawn Mth counterparty in multiple runs. Instead, we make use of the risk contributions VaR_i on counterparty level provided by the credit portfolio model (cf. Sect. 2). Using this approach, the model results for a fixed portfolio of M counterparties yield a data set of M observations consisting of the idiosyncratic parameters of a particular counterparty and its risk contribution.

To reduce the intrinsic error of the credit portfolio model, the resulting VaR_i are averaged over several Monte Carlo runs as described in Sect. 4.1.

4.3.1.2 Data preparation In this use case, the input labels are the PD, the EAD and the business sector of a single counterparty and the output label is the risk contribution VaR_i of that counterparty (we will suppress the index i in the following).

The VaR is transformed into the VaRW (4) and for each business sector VaRW is centralized around 0. Then a normalization over all business sectors is applied to the centralized VaRW, so that the resulting values lie in a range between -1 and 1.

We transform the EAD and PD in a similar fashion. At first, the logarithm of the values is centralized around 0, and afterwards a normalization is applied to the results to fall into the range between -1 and 1.

4.3.1.3 Training weights Running the ANN first without a training weight, we observe a higher RSMPE for counterparties in a sparsely populated rating class and business sector combination. Since the EAD is generated using a log-normal distribution (see Sect. 4.1), the peak for the EAD is 10^6. To reduce the influence of these highly populated areas during the training of the ANN and therefore the RMSPE, we introduce a training weight.

The weight consists of two parts. First, we adjust the contribution for one counterparty by the number of counterparties with similar EAD. To achieve this the training data is divided into 50 EAD-buckets of constant width. The effect of each training sample on the learning is reduced by a factor proportional to the number of samples in the bucket it belongs to. Second, to not overemphasize the tail of the distribution, we introduce a linear downscaling by the term $-10^{-9}\mathrm{EAD}$.

The combination of the two parts and the scaling term of the second part are determined experimentally, resulting in the training weight for a sample in bucket $b \in \{1, \ldots, 50\}$:

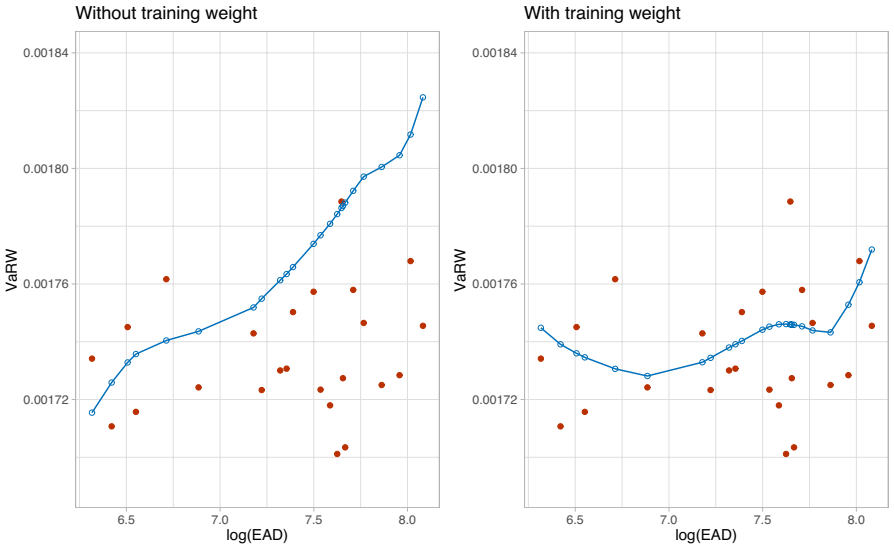

Fig. 3 Characteristic curve for rating class 1 and business sector 2, line is the prediction, data points are the counterparties in the portfolio

Table 3 Setup of the ANN and credit portfolios

Parameter	Value
Number of layers	6
Neurons per layer	64 - 64 - 64 - 64 - 64 - 1
Activation function	eLu - eLu - ReLu - ReLu - ReLu - Linear
Optimizer	Adam
Learning rate	10^{-4}
Training weight	$1 - 10^{-9}$EAD– scaling factor for data density
Maximum number of epochs	5000
Input label	Transformed EAD, PD and business sector of the counterparty
Output label	Transformed VaR of the counterparty
Number of portfolios	1
Number of counterparties per portfolio	20,000

$$1 - \frac{C_b}{\max_j C_j} \times 0.45 - 10^{-9}\text{EAD}$$

where C_b denotes the number of counterparties in bucket b. With the help of training weights, the RMSPE could be reduced from 2.74% to 1.29% in the case shown in Fig. 3.

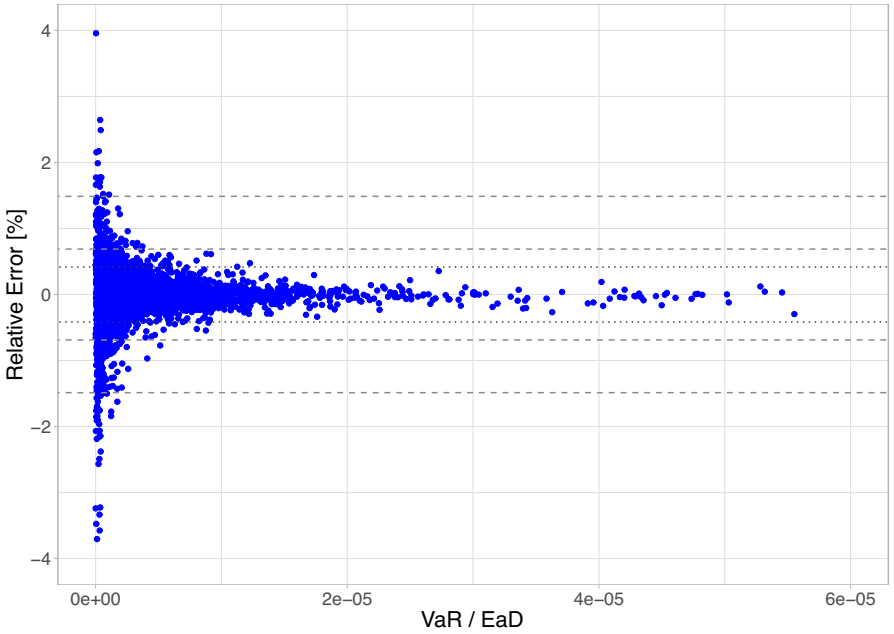

Fig. 4 Relative error for the ANN prediction, dotted line is the RMSPE, dashed lines are 95%- and 99% -confidence intervals, data points are the counterparties in the portfolio

Fig. 5 Characteristic curve for rating classes, line is the prediction, data points are the counterparties in the portfolio

4.3.1.4 Setup of the ANN Like in the use case in Sect. 4.2 we varied several parameters of the ANN. Based on the transformations of EAD, VaR and the data density described in the previous paragraphs, the setup of the ANN with the best fit is summarized in Table 3.

4.3.1.5 Evaluation The evaluation of the trained ANN with the test data set resulted in a RMSPE of 0.42% (cf. Fig. 4) for the counterparties in the portfolio.

As can be seen in Fig. 4, the relative error is highest for counterparties with small values of VaR. Since the VaR roughly scales with the rating class (lower rating classes correspond to lower VaR), this translates into lower (better) rating classes contributing most to the relative error. This is due to two effects. First, these rating classes are sparsely populated resulting in few training data points in this VaR range. Second, good rating classes are characterized by few defaults, which increases the variance of the VaR training data from the Monte Carlo simulation. Both effects make it hard for the ANN to achieve a sufficiently good approximation for low VaR. The RMSPE for the approximation of sparsely populated rating class 1 and business sector 2 is 1.29%, whereas for the more populated rating class 10 and the same business sector it is 0.27% (cf. Fig. 5).

The business sector also has an impact on the approximation error: The lower the correlation between the counterparties in the sector, the higher is the approximation error. The highly dependent business sector 1 has a RMSPE of 0.18%, sector 2 has a RMSPE of 0.31% and sector 3 with the lowest dependency has a RMSPE of 0.63%. This can be explained as follows: In the case of higher dependency (i.e. R^2 value), the information of the business sector (which is a model input) accounts for a greater share of creditworthiness-volatility (i.e. evolution of B_i). In case of a lower dependency, the information about the business sector is less valuable to the model, leading to a higher approximation error.

4.3.2 Approximation using quadratic regression

Next, we solve the same task using an approximation model that relies on non AI-based methods. Since the number of input labels is significantly smaller than in the first use case, a wider range of methods become feasible. It turns out that a quadratic approximation suggested by the following heuristic considerations is sufficient to surpass the results of the neural network.

- For a given rating class and a given business sector, the VaR is proportional to the EAD if the EAD is sufficiently small (i.e., as long as concentration effects are negligible).
- The factor of proportionality is a function of the rating class and the business sector.
- With higher values of EAD, the VaR increases faster than linearly, but the deviations from linearity are still small if the ratio of the counterparty's exposure to the total exposure of the portfolio is small.

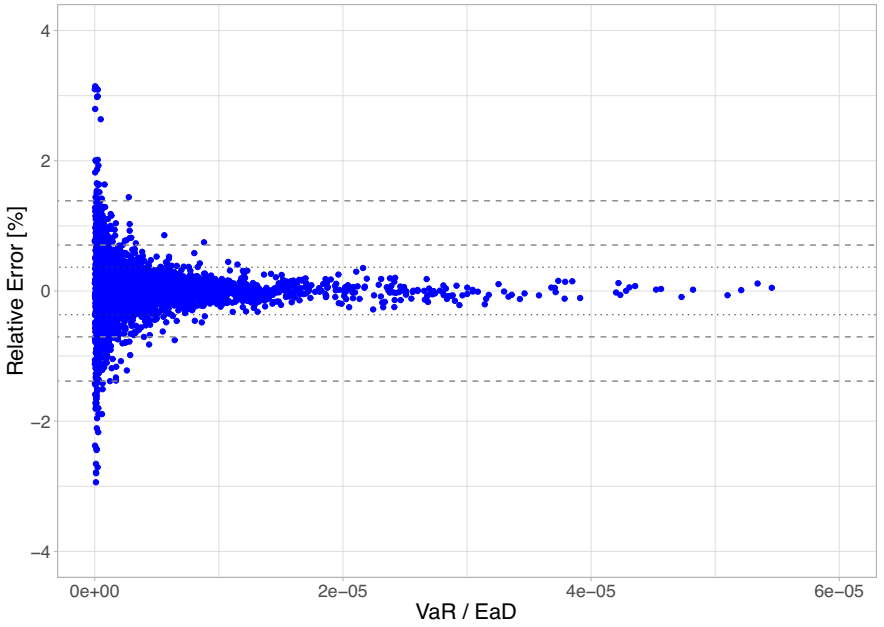

Fig. 6 Relative error for the quadratic regression, dotted line is the RMSPE, dashed lines are 95%- and 99% confidence intervals

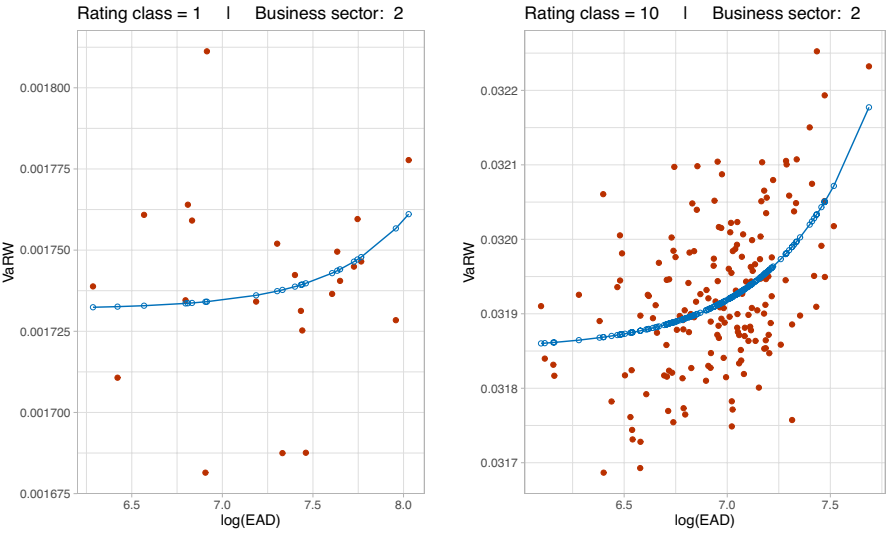

Fig. 7 Characteristic curve for rating classes, line is the prediction, data points are the counterparties in the portfolio, data points are the counterparties in the portfolio

Table 4 Comparison of the ANN and the quadratic regression

	ANN (%)	Quadratic regression (%)
RMSPE	0.42	0.37
95% confidence interval	0.69	0.71
99% confidence interval	1.49	1.38

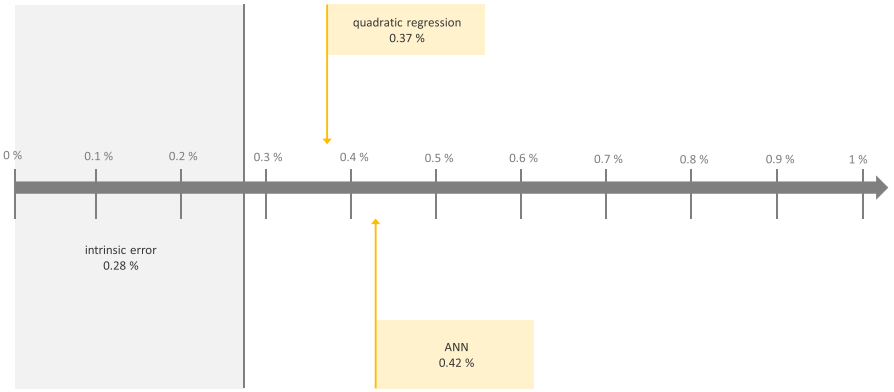

Fig. 8 Overview of approximation results from AI-based and non-AI based models

A suitable approach using quadratic regression would therefore be

$$\mathrm{VaRW} = a\left(1 + q\,\frac{\mathrm{EAD}}{\sum_j \mathrm{EAD}_j}\right), \tag{5}$$

where a and q are functions of the rating class and business sector to be determined by linear regression.

Results of this approximation are shown in Fig. 7 below. Training and testing this quadratic regression with the same data set used for the ANN in the previous sections results in an RMSPE of 0.37% (cf. Fig. 6).

4.3.3 Discussion

Both the ANN and the quadratic regression (cf. Table 4 and Fig. 8) lead to similarly low errors for the approximation of a counterparty's VaR contribution to the overall portfolio. The maximal relative error of the quadratic regression is lower compared to the ANN because the non AI-based approach can cope better with the sparsely populated rating classes. Therefore, the regression method performed better not only with respect to the overall approximation error but also regarding runtime.

The approximation error of both methods is mainly limited by the intrinsic error resulting from the Monte Carlo simulation, which is around 0.28%. The non

Table 5 Parameters for Vasicek distribution

Label	Mean	Variance
Sector 1	0.3	0.01
Sector 2	0.15	0.01
Sector 3	0.05	0.01
Inter-sector correlation	0.25	0.01

Table 6 Setup of the ANN and credit portfolios

Parameter	Value
Number of layers	6
Neurons per layer	6 - 8 - 6 - 6 - 4 - 1
Activation function	ReLu - ReLu - ReLu - ReLu - ReLu - Linear
Optimizer	Adam
Learning rate	10^{-3}
Learning weight	1
Maximum number of epochs	1000
Input label	Inter- and intra-sector correlation for the portfolio
Output label	Transformed VaR of the portfolio
Number of portfolios	1500
Number of counterparties per portfolio	1000

AI-based approximation assumes a simple functional dependence between EAD and VaR, which—in the case of our portfolio setup—captures the actual situation quite accurately. In the regime of sparse training data, this gives a clear advantage over the (model-free) ANN (cf. Fig. 7).

4.4 Use case 3: predicting model parametrization effects on total VaR

In the previous use cases, we used an ANN to predict the overall or counterparty VaR as a function of changing portfolio or idiosyncratic risk parameters (i.e., EAD, PD) whereas the systematic risk parameters (i.e. dependency parameters) were held constant. In times of an economic recession or due to other circumstances, these parameters can also vary. Therefore, portfolio models need to be validated frequently, which may lead to updated dependency parameters. In this section, we investigate the capability of an ANN to predict the portfolio VaR with respect to changing model parameters, while the underlying portfolio is constant. We will focus on changing the dependency parameters, i.e.:

- dependency between counterparties and business sectors (intra-sector dependency determined by R_k)

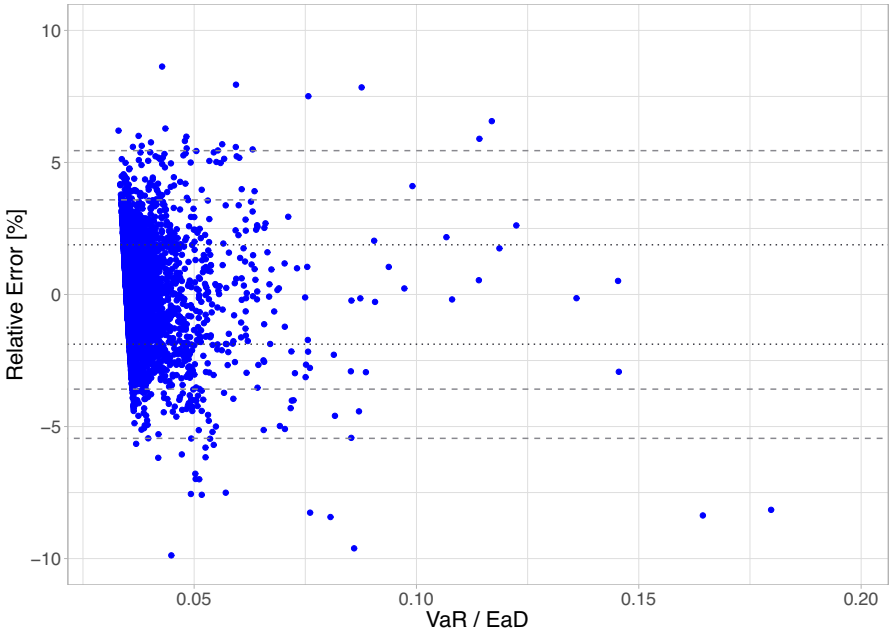

Fig. 9 Relative error for ANN, dotted line is the RMSPE, dashed lines are 95%- and 99% confidence intervals, data points are the different configurations of the portfolio

- dependency between the different business sectors (inter-sector dependency, determined by the sector copula and its dispersion matrix Σ)

4.4.1 Approximation using an ANN

4.4.1.1 Creation of the portfolios Since the focus of this use case is the impact of the dependency parameters, the counterparties within the portfolios should be constant. Therefore, we create one master portfolio defining the parameters of each counterparty and used it with different parameter sets for the dependency parameters.

During creation of the master portfolio, we include a concentration risk as described in Sect. 4.1 within the VaR-calculation to provoke stronger non-linear relationships between counterparties' EAD and their VaR contribution.

4.4.1.2 Drawing sector correlations We draw the inter- and intra-correlation of sectors from Vasicek distributions with parameters shown in Table 5. We choose the mean to be the fixed value described in Sect. 4.1.

4.4.1.3 Transformation of VaR For better training performance, the VaR of the training portfolios is divided by the portfolio EAD before feeding it into the ANN. We

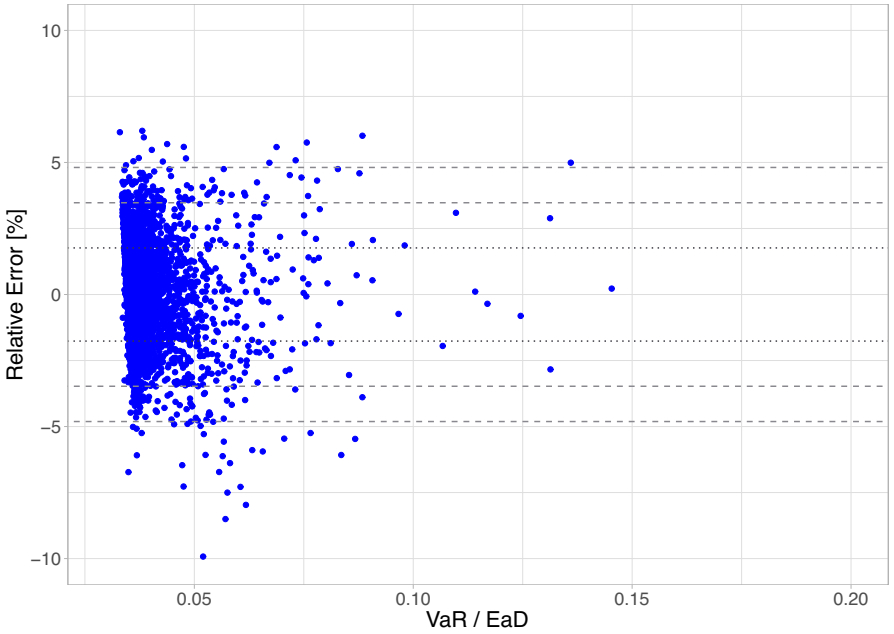

Fig. 10 Relative error of a fourth-order Taylor approximation, dotted line is the RMSPE, dashed lines are 95%- and 99% confidence intervals, data points are the different configurations of the portfolio

Table 7 Comparison of the ANN and the Taylor approximation

	ANN	4th order Taylor approximation
RMSPE	1.88%	1.77%
95% confidence interval	3.58%	3.47%
99% confidence interval	5.45%	4.87%

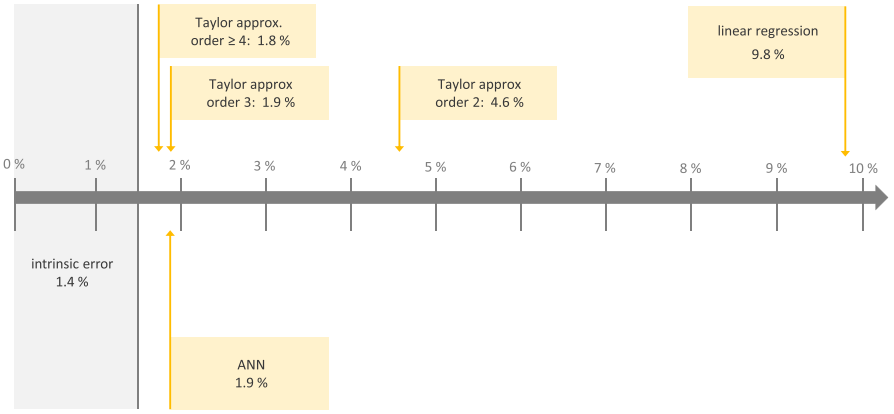

Fig. 11 Overview of approximation results from AI-based and non-AI based models

do not perform any further transformation since the input labels, i.e. the correlation parameters, are already all between 0 and 1.

4.4.1.4 Setup of the ANN The best setting of parameters we could find for the ANN is described in Table 6.

4.4.1.5 Evaluation Evaluating the ANN with the test data set results in a RMSPE of 1.88% (cf. Fig. 9).

There is no obvious relationship between the VaR resulting from different correlation parameter configurations and the quality of the approximation. There are configurations outside the 95% confidence interval over the complete VaR spectrum.

4.4.2 Taylor approximation

As an alternative non AI-based method to solve the approximation task, we use a non-linear regression based on the Taylor approximation of the ratio of the total VaR to the total EAD. This is an appropriate method here as this ratio varies slowly with model parameterization and the number of model parameters is small.

The input labels x consists of 3 intra-sector dependencies and 3 independent components of the dispersion matrix, $x = (R_1, R_2, R_3, \Sigma_{12}, \Sigma_{13}, \Sigma_{23})$. We use the following approach

$$\mathrm{VaR} = \mathrm{EAD} \cdot \sum_{|a| \leq g} \alpha_a \, (x - \bar{x})^a \,, \tag{6}$$

where a denotes a 6-dimensional multi-index, $\bar{x}$ the mean of x over the training data and g the order of the approximation. The regression coefficients α (for a given order g, α has $(6^{g+1} - 1)/5$ components) are determined by minimizing the mean squared error on the training data.

We evaluate this regression for different orders on the test data set (cf. Fig. 11). Up to fourth order, the RMSPE decreases. For higher orders, the gain in precision is neglectable. Overall, we achieve a RMSPE of 1.77% (cf. Fig. 10) with a fourth-order approximation.

4.4.3 Discussion

For our parameter selection, both ANN and Taylor approximation (cf. Table 7 and Fig. 11) lead to similarly low approximation errors.

Again, the minimal error of both models is limited by the intrinsic error due to the VaR Monte Carlo simulation.

The training and optimizing efforts are much lower for the Taylor approximation than for the ANN. Thus, in a scenario with a small number of parameters it can be more favorable to use a non AI-based method. However, this may change with an increasing number of parameters.

5 Summary

In this paper, we compare the performance of different (AI- and non AI-based) approximation methods to speed up credit portfolio calculations. For the hypothetical portfolios used, non AI-based methods (Taylor approximation or quadratic regression) roughly match the performance of AI methods. In addition, non AI-based methods require less training data and effort and are more transparent.

The availability of training data often restricts the use of AI models. In the case of credit portfolio models, this problem is especially serious because of the high dimensionality of the input data (usually on individual loan level). In the first use case presented in this paper, we therefore employ a custom-made pre-processing (binning) approach to limit the training data requirements. Subsequently, we focus our work on two practically relevant cases that use a fixed loan portfolio as a point of reference, which massively reduces the need for training data. Once successfully trained to perform the approximation, AI models have the advantage that results can be obtained extremely quickly, and that the approximation is able to capture a wide range of non-linear relationships between input and output variables.

In summary, we demonstrate that there is a wide range of methods that can be used to obtain quick approximations for credit portfolio calculations if certain levels of approximation error can be accepted. Both AI and non AI-based methods perform well on our use cases and (hypothetical) sample portfolios. Therefore, our recommendation for the use cases considered here is to start with non AI-based methods even for larger or more concentrated portfolios; AI models should only be used where non AI-based methods fail.

Appendix A: Copulas

According to Sklar's theorem, a multivariate distribution function F on $\mathbb{R}^d$ with marginal functions F_i can be decomposed with the help of a unique copula function

$$C : \times_{i=1}^{d} \mathrm{Im}(F_i) \to [0,1] \text{ such that}$$
$$F(x) = C(F_1(x_1), ..., F_d(x_d)) \text{ for all } x \in \mathbb{R}^d.$$

The t-copula is the copula function implicitly defined by the multivariate t-distribution with a given dispersion matrix Σ and ν degrees of freedom. For more details on copula functions and risk management, we refer to McNeil et al. (2015).

Appendix B: Detailed overview of methods and results of use case 1

Table 8 Additional methods for prediction of the total VaR

AI Method	CUBIST	SVR Radial	SVR Linear	XGBoost
RMSPE	3.91%	4.18%	4.43%	4.98%
Max. outlier	47.5%	47.8%	22.5%	42.0%
Number of portfolios	63000	63000	15000	63000
Input labels	$C_b, C_b \cdot E_b, C_b \cdot P_b$	$C_b, C_b \cdot E_b, C_b \cdot P_b$	$C_b \cdot E_b, C_b \cdot P_b$	$C_b, C_b \cdot E_b$

Table 9 Additional methods for prediction of the total VaR

AI Method	SVR 3rd degree polynomial kernel	Random Forest™	MARS	CART	K Nearest Neighbor
RMSPE	5.77%	6.79%	9.82%	11.0%	11.9%
Max. outlier	> 100%	48.1%	48.0%	49.6%	37.6%
Number of portfolios	63000	31000	31000	31000	63000
Input labels	$C_b \cdot E_b$	$C_b \cdot E_b$	$C_b \cdot E_b$	$C_b \cdot E_b, C_b \cdot P_b$	$C_b \cdot E_b$

Variables used for the input labels: transformed number of counterparties C_b, transformed EAD E_b, for definition and details see 4.2.1 and transformed probability of default P_b, derived from binning in the same way as the other variables.

Maximal outliner is the highest relative error of a single test sample that occurred in the predictions of the test data set (70/30 train/test split applied).

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability The data underlying the findings are simulated. Therefore they cannot be shared. Information regarding the data generating process are given within the article. If necessary, more information are available upon request.

References

Akhavein, J.D., Kocagil, A.E., & Neugebauer, M. (2005). A comparative empirical study of asset correlations. *Fitch Ratings, New York*

Aziz, S., & Dowling, M. (2019). Machine learning and AI for risk management. In: *Disrupting Finance: FinTech and Strategy in the 21st Century*, pp. 33–50

Bhaskar Sundaram, R. (2022). *An End-to-End Guide to Understand the Math behind XGBoost*. Retrieved 10 Nov 2022 from https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

Breiman, L. (1984): *Classification And Regression Trees (1st ed.)*. Routledge. https://doi.org/10.1201/9781315139470

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chen, T., & Guestrin, C. (2016). *Xgboost: A scalable tree boosting system*. CoRR abs/1603.02754

Chun, S. Y., & Lejeune, M. A. (2020). Risk-based loan pricing: Portfolio optimization approach with marginal risk contribution. *Management Science, 66*(8), 3735–3753. https://doi.org/10.1287/mnsc.2019.3378

Credit Swiss First Boston. (1997). *Creditrisk+: A credit risk management framework*. Technical report, Credit Suisse First Boston: Technical report.

Dixon, M.F., Halperin, I., & Bilokon, P. (2020). *Machine Learning in Finance vol. 1170*. Berlin: Springer

Dorfleitner, G., Fischer, M., & Geidosch, M. (2012). Specification risk and calibration effects of a multifactor credit portfolio model. *The Journal of Fixed Income, 22*(1), 7–24. https://doi.org/10.3905/jfi.2012.22.1.007

Duffie, D. (2003). *Credit risk: Pricing, Measurement, and management*. Princeton series in finance. Princeton: Princeton University Press

Düllmann, K., Küll, J., & Kunisch, M. (2010). Estimating asset correlations from stock prices or default rates—which method is superior? *Journal of Economic Dynamics and Control, 34*(11), 2341–2357. https://doi.org/10.1016/j.jedc.2010.06.003

Geidosch, M. (2014). Asset correlation in residential mortgage-backed security reference portfolios. *Journal of Credit Risk 10*(2)

Glasserman, P. (2004). Monte Carlo Methods in financial engineering. *Stochastic Modelling and Applied Probability*. https://doi.org/10.1007/978-0-387-21617-1

Gordy, M., & Heitfield, E. (2010). Small-sample estimation of models of portfolio credit risk. *Recent Advances in Financial Engineering, 2009*, 43–63. https://doi.org/10.1142/9789814304078_0002

Grundke, P., & Moosbrucker, T. (2008). Approaches to generate the loss distribution. In: *The Definitive Guide to CDOs*, pp. 161–185

Gupton, G. M., Finger, C. C., & Bhatia, M. (1997). *CreditMetrics: Technical document*. New York: J.P. Morgan & Co.

Hahnenstein, L. (2004). Calibrating the CreditMetrics correlation concept-Empirical evidence from Germany. *Financial Markets and Portfolio Management, 18*(4), 358–381.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(8), 832–844. https://doi.org/10.1109/34.709601

Huang, J., Chai, J., & Cho, S. (2020). Deep learning in finance and banking: A literature review and classification. *Frontiers of Business Research in China, 14*(13). https://doi.org/10.1186/s11782-020-00082-6

Jacobs, M. J. (2010). An empirical study of exposure at default. *Journal of Advanced Studies in Finance, 1*(1), 31–59.

Jakob, K., & Fischer, M. (2016). GCPM: A flexible package to explore credit portfolio risk. *Austrian Journal of Statistics, 45*(1), 25–44. https://doi.org/10.17713/ajs.v45i1.87

Kalkbrener, M., & Onwunta, A. (2010). *Validating structural credit portfolio models. Model risk—identification, measurement and management*. London: Risk Books, pp. 233–261

Kuhn, M., & Quinlan, R. (2022). Cubist: rule- and instance-based regression modeling. https://topepo.github.io/Cubist/

Lee, S.-C., Lin, C.-T., & Yang, C.-K. (2011). The asymmetric behavior and procyclical impact of asset correlations. *Journal of Banking & Finance, 35*(10), 2559–2568.

Leo, M., Sharma, S., & Maddulety, K. (2019). Machine learning in banking risk management: A literature review. *Risks*. https://doi.org/10.3390/risks7010029

Liu, S., Oosterlee, C.W., & Bohte, S.M. (2019). Pricing options and computing implied volatilities using neural networks. Risks *7*(1). https://doi.org/10.3390/risks7010016

McNeil, A. J., Frey, R., & Embrechts, P. (2015). *Quantitative risk management: Concepts*. Techniques and Tools-revised Edition: Princeton University Press.

Pfeuffer, M., Nagl, M., Fischer, M., & Rösch, D. (2019). Parameter estimation, bias correction and uncertainty quantification in the vasicek credit portfolio model. *Journal of Risk, 22*(4), 1–29. https://doi.org/10.21314/JOR.2020.429

Premanand, S. (2022). The A-Z guide to Support Vector Machine. Retrieved 14 Oct 2022 from https://www.analyticsvidhya.com/blog/2021/06/support-vector-machine-better-understanding/

Sklar, A. (1959). Fonctions de repartition a n dimensions et leurs marges. *Publ. Inst. Stat. Univ. Paris, 8*, 229–231.

Sruthi, E.R. (2022). Understanding Random Forest. Retrieved 10 Nov 2022 from https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

Turing, A.M., & Haugeland, J. (1950). *Computing machinery and intelligence.* In: Shieber, Stuart M (2004) *The Turing Test: Verbal Behavior as the Hallmark of Intelligence,* Mit Press, pp. 67–96.

Vasicek, O. A. (2002). The distribution of loan portfolio value. *Risk, 15*, 160–162.

Authors and Affiliations

Kevin Jakob[1] · Johannes Churt[2] · Matthias Fischer[3] · Kim Nolte[2] · Yarema Okhrin[1] · Dirk Sondermann[2] · Stefan Wilke[2] · Thomas Worbs[2]

✉ Kevin Jakob
kevin.jakob.research@gmail.com

Johannes Churt
johannes.churt@basycon.com

Matthias Fischer
Matthias.Fischer@fau.de

Kim Nolte
kim.nolte@basycon.com

Yarema Okhrin
yarema.okhrin@wiwi.uni-augsburg.de

Dirk Sondermann
dirk.sondermann@basycon.com

Stefan Wilke
stefan.wilke@basycon.com

Thomas Worbs
thomas.worbs@basycon.com

[1] Department of Statistics, University of Augsburg, Universitätsstraße 16, Augsburg 86159, Bavaria, Germany

[2] Basycon Unternehmensberatung GmbH, Welserstraße 1, Munich 81373, Germany

[3] Department of Statistics & Econometrics, Friedrich-Alexander-Universität Nürnberg, Lange Gasse 20, Nuremberg 90403, Bavaria, Germany