The 7th International Conference on Economics and Social Sciences
Exploring Global Perspectives:
The Future of Economics and Social Sciences
June 13-14, 2024
Bucharest University of Economic Studies, Romania

Using Machine Learning to Model Bankruptcy Risk in Listed Companies

Vlad TEODORESCU[1*], Cătălina-Ioana TOADER[2]

DOI: 10.24818/ICESS/2024/055

Abstract

This article extensively studies the optimisation and relative performance of three classes of machine learning models (logistic regression with regularisation, Random Forest, and XGBoost) to quantify the probability of bankruptcy using financial data from a database of listed companies in Taiwan. The database covers the period from 1999 to 2009, contains 95 financial ratios from 7 categories, has 6,819 observations, and has a bankruptcy rate of approximately 3.2%. The database choice stemmed from our wish of utilising a dataset which was publicly available and that posed high quality and moderate size, traits that permitted the rapid training of machine learning models. As a result, we were able to run experiments based on multiple model configurations and to compare the attained results with the ones gathered by other researchers. For the purpose of splitting data for training and testing sets, the k-fold cross-validation methodology can be used. We investigate the validity of its use, especially in the context of XGBoost with an early stopping round based on the test fold. We also determine the sensitivity of predictive performance on the value of k and on the specific folds created. We use AUROC as a performance measure and show that Random Forest models significantly outperform logistic models with regularisation, while XGBoost models have a moderately higher performance than Random Forest. For each type of model, we study hyperparameter tuning and demonstrate that this process has a significant effect on predictive performance. For the first two types of model, we perform a full grid search. For XGBoost models, we use a guided (sequential) grid search methodology. Furthermore, we study and propose a criterion for hyperparameter tuning using average performance instead of maximum performance, highlighting the relatively large effect on predictive performance of the stochastic component employed by these machine learning algorithms during training. Our research also indicates that in the case of some hyperparameters, tuning can shape predictive performance. Last but not least, the meaningfulness of variables in forecasting the bankruptcy likelihood is assessed, as it was indicated by the three classes of models.

Keywords: bankruptcy risk, probability of bankruptcy, machine learning, xgboost, random forest.

[1] Bucharest University of Economic Studies, Bucharest, Romania, vlad.teodorescu@fin.ase.ro.
* Corresponding author.
[2] Bucharest University of Economic Studies, Bucharest, Romania, catalina.toader@fin.ase.ro.

JEL Classification: C53, C55, D81, G2, G32.

1. Introduction

In the last years, artificial intelligence displayed a significant development level, altogether with the emergence of deep learning and GPT models. The drawback resides however in the fact that this type of models need large amounts of data (hundreds of millions or even billions of observations). As a result, machine learning models (such as random forests or boosted trees) remain relevant for the comparatively small datasets used for bankruptcy prediction.

Although there are a number of studies that use and compare various machine learning algorithms for bankruptcy prediction, very few aim to find the best possible model by studying various architectures and hyperparameter tuning.

In this paper, we search for the best model we can find and, in the process, make recommendations and present findings relevant in guiding such a search: from splitting the data into training and test sets to hyperparameter tuning strategies based on model type.

In the whole process, we trained 1,062 logistic regression models, 17.646 Random Forest models, and 881 XGBoost models.

2. Problem Statement

Lending institutions and investors are exposed on a daily basis to challenges resulting from the risk of adverse events such as default or bankruptcy. A resolution point could be artificial intelligence (AI), which, through the employment of large data and machine learning, boosts risk evaluation. In addition, it contributes to more informed financing decisions based on the analysis of borrowers' financial data, that can ultimately diminish default and bankruptcy rates and increases capital flow in the financial industry. Liang et al. (2016) examine the effectiveness of integrating financial ratios (FRs) and corporate governance indicators (CGIs) for bankruptcy prediction with real-world data from Taiwan, using machine learning methods. They found that FRs and CGIs together improve prediction accuracy. In the realisation of a precise projection, elements such as profitability and solvency FR categories are key points, apart from board and ownership structure CGIs.

Lately, many researchers have forecast loan defaults using machine learning methods. In a recent study, Lin (2024) used four machine learning algorithms, including Logistic Regression, Random Forest, XGBoost and AdaBoost, underscoring the efficacy of the Gradient Boosting framework, particularly the XGBoost model, in predicting loan defaults. Although it may have some limitations, this research revealed that while maximising loan profitability is paramount in default prediction, the analysis did not extend to calculating final payoffs predicted by different algorithms. Moreover, the author suggests considering factors like loan interest rates to more accurately assess final payoffs. In addition to that, Hernes et al. (2023**)** stated that XGBoost is an intriguing possible substitute for the conventional process of making a score card. The authors suggested that the implementation of sophisticated algorithms in credit decision-making procedures will necessitate modifications to the IT infrastructure of numerous institutions, perhaps resulting in extra financial outlays, but these algorithms may actually have a significant effect on how well decisions are made.

Per Moscatelli et al. (2019), in their effort of assessing potential borrowers, lenders or credit analysts can find it advantageous to employ statistical and machine learning models in a combined approach. Additionally, Sifrain (2023) emphasises that recovery, debt-to-income ratio, annual income and loan amount, as independent variables, reflected to be correlated in

the most significant manner with the response variable, especially when it comes to the logistic regression and random forest classifiers.

In their research in which they employed a Random Forest and Decision Trees method to establish a model for loan forecasting and credit risk evaluation, Madaan et al. (2021) revealed that a higher degree of precision was provided by the first model. In addition, Alonso and Carbó (2021) note that machine learning models operate better compared to the conventional Logit model, in the classification and calibration tasks. Additionally, the authors acknowledge these facts through sensitivity analysis in a different setting. Last but not least, Hyeongjun et al. (2020) notes that it is important to understand the manner in which every approach should be used when establishing the proper technique which may generate relevant data for the prediction goal.

3. Research Questions / Aims of the Research

Our main objective is to discover a methodology for searching the best-performing model given a dataset. In the process, we answer several questions: (1) Can the k-fold cross-validation methodology be safely used with XGBoost? (2) What is the optimal number for *k* and how to find it? (3) How sensitive are the model performances to the specific (random) split of data in the k folds? (4) What is the relative performance of various machine learning models (logistic regression with regularisation, Random Forest, XGBoost)? (5) What is the optimal methodology for hyperparameter tuning depending on model type? (6) How sensitive are the model performances to the randomness induced by the training algorithm? In other words, using the same hyperparameters and data splits, how much does performance vary due to different random seeds used by the algorithm? The answer to this question will guide the choice of optimal hyperparameter values: should we base our choice on the best model or on average performance? (7) What are the most important predictors of bankruptcy? (8) Do the most important predictors differ for different model types?

4. Research Methods

For comparability, we opted to use a publicly available dataset. We investigated several options and settled for the dataset compiled by Liang et al. (2016). We chose this dataset due to (1) the large number of predictors (95), some correlated, suitable for investigating various machine learning algorithms, (2) its relatively small size for machine learning models (6,819 observations), which allowed us to efficiently test alternative methodologies, and (3) its quality (having little noise and no missing data). The data set is comprised of Taiwanese listed companies, has 95 financial ratios as explanatory variables, and a bankruptcy rate of 3.22%, defined according to the Taiwan Stock Exchange rules for bankruptcy. The data covers the period 1999-2009 and consists of non-financial companies that had at least three years of available data.

For testing, we used **the k-fold cross-validation** approach, which involves randomly partitioning the data into k equally sized folds and then iteratively setting aside one fold for out-of-sample testing and training a model on the remaining k-1 folds. In total, for each model, k component models will be trained. When using the model with new data for prediction, the average of all k component models' predictions will be considered. As a result, the whole dataset will be used for training and out of sample predictions will be obtained for the entire training dataset.

In creating the k folds, we used stratified sampling based on the target variable, hence ensuring equal bankruptcy rate across all folds.

We studied three model types: **logistic regression with regularisation, Random Forests, and XGBoost**. We trained 19,589 individual models to identify the best-performing model type and also to tune its hyperparameters.

Logistic regression with regularization has a loss function with an additional term compared to standard logistic regression:

$$\hat{\beta} = \arg\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - x_i^T \beta)^2 + \lambda \left(\alpha \sum_{j=1}^{p} |\beta_j| + \frac{1-\alpha}{2} \sum_{j=1}^{p} \beta_j^2 \right) \right\} \quad (1)$$

The first part of the formula corresponds to the OLS method. The effect of the regularisation term is to minimise the coefficients β, which controls the risk of overfitting. There are three types of regression with a regularisation term: Lasso – for α=1, Ridge – for α=0 and Elastic net – for α between 0 and 1.

The hyperparameter λ controls the effect of the magnitude of the regularisation term's effect.

Random forests (RF) imply the training of multiple decision trees and taking the average of their prediction. Each decision tree is trained on a subset of data, and at each node, a subset of features is selected. As a result, each decision tree is suboptimal and captures one aspect of the solution. However, averaging across multiple decision trees will improve performance beyond what the best single tree could do.

XGBoost (eXtreme Gradient Boosting) is an algorithm that, like Random Forest (RF), is based on decision trees. Unlike RF, which builds independent trees unconnected to each other, XGBoost iteratively constructs trees, each one attempting to correct previous errors. With each iteration, the newly constructed tree is trained on the residuals of the previous ones and not on the original Y variable. In this way, the predictions are improved with each iteration. If training is not stopped, the algorithm will overfit the training data to a high degree. As a result, it is customary to use a test set to stop the training once out-of-sample performance does not improve for several iterations, typically in the order of hundreds.

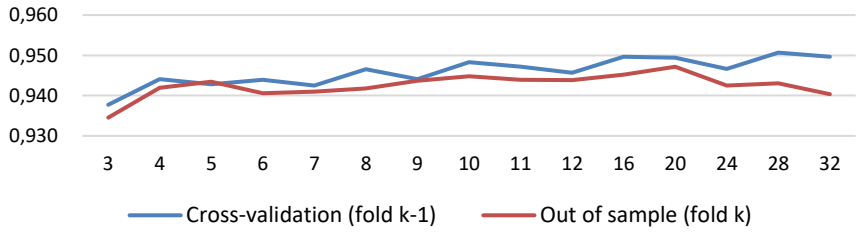
To assess feature importance across three model classes, we use **permutation feature importance**, since it is a model-independent method. The importance is calculated as the loss in model performance when its values are permuted between observations in a test set.

5. Findings

5.1 Assessing the Validity of k-fold Cross-Validation for XGBoost

With the k-fold cross-validation approach for XGBoost, the test fold will also be used during model training to determine the best iteration and stop the algorithm. As a result, there is a risk of label leaking.

Figure 1. Performance based on k, the number of folds



Source: own calculations.

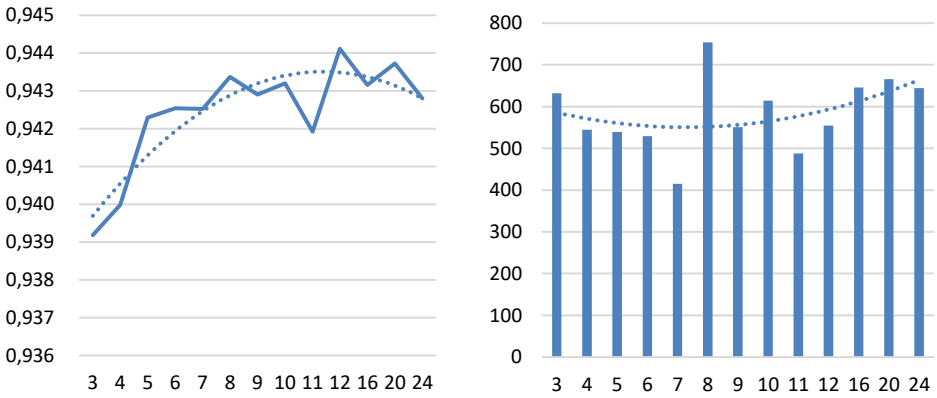
To assess the validity of using the k-fold cross-validation technique with XGBoost, we conducted an analysis with hierarchical cross-validation: each component model was trained using k-2 folds, the k-1 fold was used to determine the stopping point, and the kth fold was used for out-of-sample testing. We observed that (1) the performance differences between the k-1 and k folds are minimal and, more importantly, that (2) the performances on these two folds are highly correlated (correlation coefficient of 0.698). As a result, we concluded that the k-fold cross-validation technique remains viable.

5.2 Determining the Optimal k

The larger k, the more data can be used for model training, thus leading to better performance. Indeed, as Figure 1 shows, up to a point, larger ks produce higher performance out of the sample. On the downside, the larger k is, the smaller each fold is, hence potentially leading to overfitting the fold used to stop the algorithm's training: as it can be seen in Figure 1, for ks above 20, the performance on the two test folds start to diverge, indicating overfitting the k-1 fold used to stop the algorithm. Another disadvantage of a large k is the time and computational resources required to train k models.

Hence, the smallest k that offers good performance is preferred. We also considered the variability of the stopping round number for each of the k component models as a measure of overfitting the k-1 fold. We have settled on k=7.

Figure 2. Indicators for choosing k – left: out of sample performance, right: standard deviation of stopping iteration



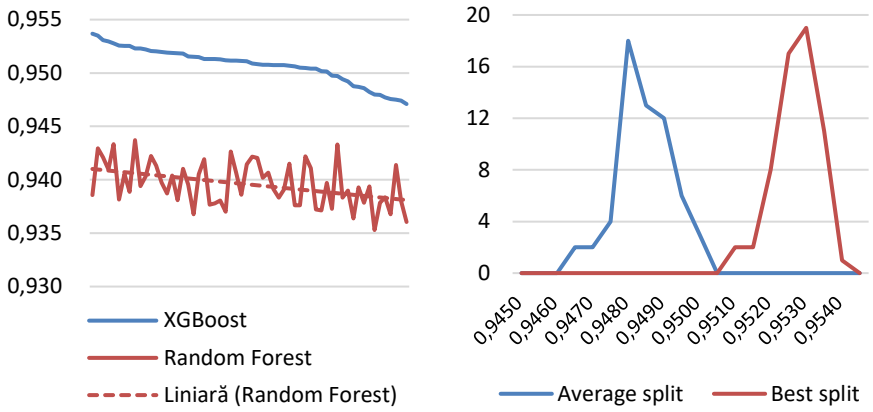
Source: own calculations.

5.3 Choosing a Specific Split for 7 Folds

We performed a random stratified sampling based on the target variable. We calculated model performance on various random splits to assess performance sensitivity. We noted that models trained on some splits outperform others. This performance gap remained even between model types (XGBoost vs. Random Forest).

We settled on the split that produced the highest performance across both types of models. To confirm the superiority of this split, we trained 60 XGBoost models on it and compared the results with 60 models trained on another split.

Figure 3. Criteria for choosing a specific split – left: XGBoost and RF model performance on 60 alternative splits, sorted by XGBoost model performance, right: performance for 60 XGBoost models on two random splits



Source: own calculations.

5.4 Logistic regression with Regularisation – Hyperparameter tuning

We used the *glmnet* library in R, which uses the gradient descend method for training, meaning that the algorithm is iterative, stochastic, and, in some cases, not convergent. There are two hyperparameters: α and λ. While α needs to be set by the user, for λ the library has an option to optimise it automatically.

The algorithm was run for 21 values of the hyperparameter α, ranging from 0 to 1, with a step of 0.05. During the use of the algorithm, it was observed that it does not always converge for the λ values that it chooses by itself. To correct this effect, λ values were manually set from the average values at which the algorithm converged. Therefore, for each of the 21 values of α, a λ value was manually set. Subsequently, 29 different values of λ were studied for each of the 21 values of α, totalling 609 models. The 29 λ values were chosen by multiplying the selected average value by a scaling factor ranging from 0.005 to 10.

The best performance was 0.9314, achieved for α=0.85 and λ=0.01004262. Of the 609 models trained, 214 had at least one component model that was not convergent, while 395 were fully convergent.

5.5 Random Forest – Hyperparameter Tuning

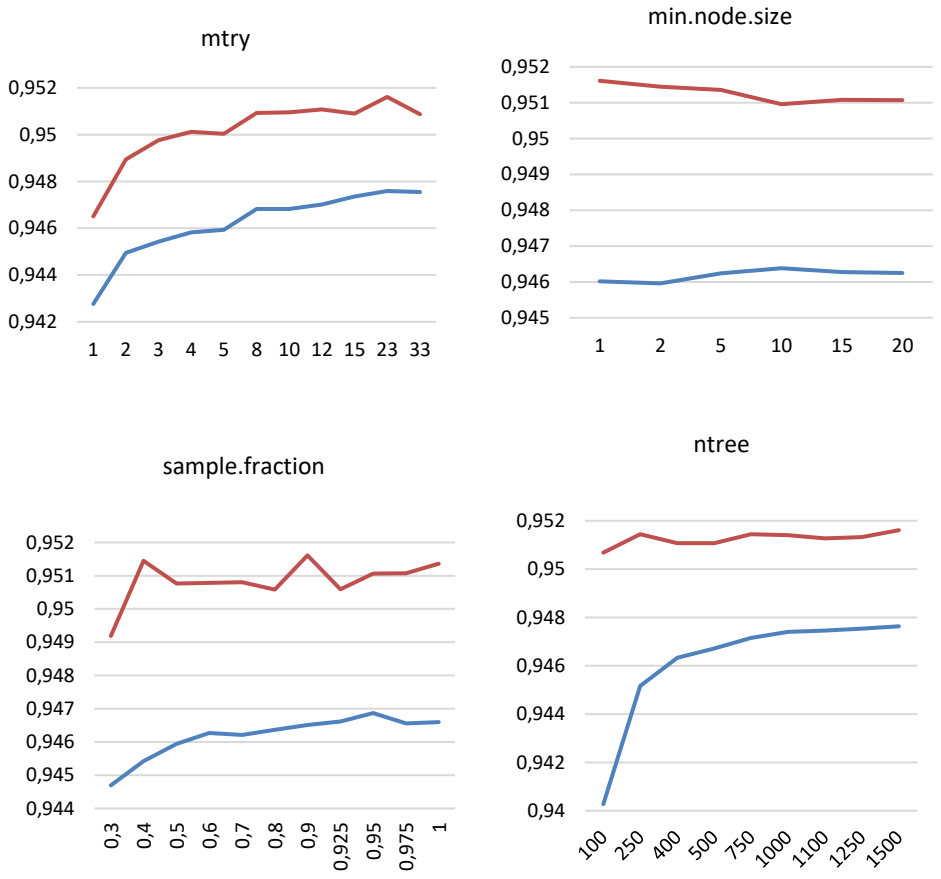
We used the *ranger* R library to train random forests. We used a full grid search methodology to tune the most important hyperparameters: *mtry*, *min.node.size*, *sample.fraction*, and *ntree*. We also studied two-node splitting rules: Gini and Hellinger (controlled by the *splitrule* hyperparameter). The Gini rule is most commonly used for classification problems. The rule based on Hellinger distance is less commonly used; however, it is more suitable for classification problems with uneven classes, as in our case here. Indeed, using the Hellinger distance significantly improved the maximum performance, from 0.9448 to 0.9516.

For hyperparameter tuning, we trained 6,534 different models using the Hellinger splitting rule and 4,356 different models using the Gini rule. As mentioned earlier,

the Hellinger method yielded significantly superior results. The best model achieved a performance of 0.9516, significantly higher than the best logit model.

When hyperparameter tuning is performed, we obtained the following results:

Figure 4. Hyperparameter tuning for Random Forests using the Hellinger splitting rule (blue: average performance, red: best performance)



Source: own calculations.

5.6 XGBoost – Hyperparameter Tunning

Due to the high computational and time resources required to train an XGBoost model and also due to the high number of hyperparameters that can be tuned, we could not perform a full grid search, as in the case for logistic regression or Random Forest. We performed a guided, iterative, grid search where we varied a few hyperparameters at one time, retained their best values, and then proceeded to vary others. We repeated this process and even came back to previously tuned hyperparameters to check whether in a new context they can be further optimised.

We also experimented with various model architectures involving (1) different objective functions, (2) winsorizing and rescaling some explanatory variables, (3) using only the most important 30 or 40 explanatory variables (as identified by a previously best performing model), or (4) including predictions from other model types (logistic regression or Random Forest) as explanatory variables.

Regarding hyperparameter tuning, we found the following:

eta: it is the most important hyperparameter since values that are too high can severely impact performance. However, below a certain point, smaller values of eta will have a negligible impact on performance. Since training time is significantly increased for lower values of eta, finding the optimal value is important. We recommend performing an initial grid search on {0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1}, followed, if necessary, by a finer search around the best value.

colsample_bytree and subsample: after eta, these two hyperparameters had the highest impact on performance. We found optimal values in the interval [0.4, 0.6]. We recommend investigating the full range of values for these hyperparameters (from 0 to 1), possibly starting with a step of 0.1 and then refining the search by a step of 0.05 around the best values identified so far.

min.child.weight: variation of this hyperparameter brought moderate improvement in average performance at the value of 5 and had a significant negative impact on performance at higher values (15 or 20).

max_depth: in our case, the impact of varying this hyperparameter was not significant. We recommend exploring various values for this hyperparameter and, all else being equal, choosing the smallest value possible as higher values can increase the training time proportionately.

alpha and lambda: Variations in these parameters beyond the default values led to performance deterioration.

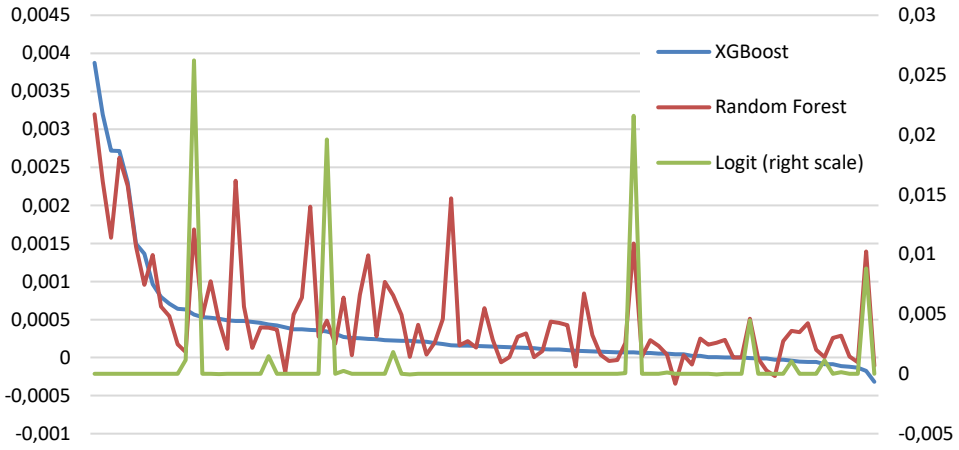
As mentioned previously, we used guided grid search, varying only some hyperparameters at a time while holding the others constant. At this point, one question arises: When moving to the next set of hyperparameters, what values should one choose for the ones just studied? Should they be based on the best model so far or on the average performance for all models trained? Taking another look at Figure 1, we can note that for models with the same architecture and the same values for all hyperparameters, the performance can vary greatly due to the randomness induced by the training algorithm. Hence, we recommend choosing values that produce the best average performance, since the best performing model could be an outlier.

5.7 Feature Importance

We calculated the feature importance for the best performing model in each class. For this purpose, we considered only models trained on the original set of features.

We found that the Random Forest models and the XGBoost models corelate to a high degree, assigning importance to the same variables. Moreover, these two types of model tend to use all variables; hence, all of them have at least some importance. In contrast, the logistic regression has only 10 important variables, the rest having an impact of almost 0.

Figure 5. Feature importance, sorted by XGBoost model's feature importance



Source: own calculations.

5.8 Performance for Different Model Types

In summary, excluding models that were obvious underperformers (for example, with wildly out of range hyperparameter values used for exploratory purposes), we obtained the following results:

Table 1. Summary of model performance by model type

Indicator	Logistic regression with regularisation	Random Forest	XGBoost
Minimum	0.8521	0.9271	0.9458
Maximum	**0.9314**	**0.9516**	**0.9543**
Average	0.9264	0.9462	0.9512
Standard deviation	0.0079	0.0031	0.0019

Source: own calculations.

6. Conclusions

We conclude that k-fold cross-validation is a valid method for training and testing when limited data is available even though some algorithms, such as XGBoost, use the test fold in the training process.

We found that Random Forests significantly outperform logistic regression with regularisation, while XGBoost outperforms Random Forests, but to a lesser degree.

We also conclude that hyperparameter tuning can have a significant impact on performance, but the randomness induced by the training algorithm is significant. As such, optimal values for hyperparameters should be chosen on average performance, not on the best model identified. Moreover, fine-tuning hyperparameters only marginally improves performance, well within the variability induced by the randomness of the training algorithm, further supporting the choice of best average performance over best performing model.

Bibliography

[1] Alonso, A., Carbó, J.M. (2021). Understanding the Performance of Machine Learning Models to Predict Credit Default: A Novel Approach for Supervisory Evaluation. Banco de Espana Working Paper No. 2105.

[2] Hernes, M. et al. (2023). Credit Risk Modeling Using Interpreted XGBoost. European Management Studies (previously: problemy zarządzania - management issues), 21(3), 46-70.

[3] Hyeongjun, K. et al. (2020). Corporate Default Predictions Using Machine Learning: Literature Review. Sustainability 12(16):6325.

[4] Liang, D. et al. (2016). Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study. European Journal of Operational Research 252(2), 561-572.

[5] Lin, J. (2024). Research on loan default prediction based on logistic regression, randomforest, xgboost and adaboost. SHS Web of Conferences.

[6] Mehul, M. et al. (2021) IOP Conf. Ser.: Mater. Sci. Eng. 1022 012042.

[7] Moscatelli, M. et al. (2019). Corporate default forecasting with machine learning, Temi di discussione (Economic working papers) 1256, Bank of Italy, Economic Research and International Relations Area.

[8] Sifrain, R. (2023). Predictive Analysis of Default Risk in Peer to-Peer Lending Platforms: Empirical Evidence from Lending Club. Journal of Financial Risk Management, 12, 28-49.