Lietuvos matematikos rinkinys
Proc. of the Lithuanian Mathematical Society, Ser. A
Vol. 65, 2025, pages 39–51
https://doi.org/10.15388/LMR.2025.44496

Vilnius University Press

Comparison of relevant credit risk assessment algorithms

Simas Rimašauskas✉, **Igoris Belovas**, **Rolandas Gricius**

Institute of Computer Science, Vilnius University, Lithuania

✉ Corresponding author: simas.rimasauskas@gmail.com
E-mail: Igoris.Belovas@mif.vu.lt; rolandas.gricius@mif.vu.lt

Received June 20, 2025; published December 22, 2025

Abstract. Assessing credit risk is essential when making financial decisions, especially investing in debt securities. As the bond market is the largest securities market in the world, a great demand exists for tools to assess issuer creditworthiness. Furthermore, information describing the probability of default is also useful in other areas, such as risk management. In the era of machine learning and big data, new techniques have emerged that allow for automated risk assessment based on large amounts of data. Traditional creditworthiness assessment methods may be inaccurate, as the investor could be biased or misinterpret available information. A review and comparison of modern tools that would allow intelligent processing of large amounts of information will help assess the issuer's credit risk as objectively as possible.

Keywords: credit risk; machine learning; XGBoost; LightGBM; AdaBoost; logistic regression; random forest
AMS Subject Classification: 68Q32, 91G40

1 Introduction

Assessing credit risk is a critical component of financial decision-making, particularly when investing in debt securities. As the bond market represents the largest segment of the global securities market, investors increasingly seek reliable tools to evaluate the creditworthiness of bond issuers with accuracy and objectivity. The emergence of machine learning and artificial intelligence has introduced more sophisticated and data-driven methods for risk assessment, enabling deeper insights based on vast and complex datasets.

When acquiring debt securities, investors must evaluate the likelihood that the issuer may default on its payments. Manual assessment may lead to a distorted understanding of credit risk, as the investor may be biased or misinterpret the information. Therefore, reviewing tools to process large amounts of information intelligently and infer the default probability could lead to a more objective issuer credit risk assessment.

This study aims to review the principles behind the key modern credit risk assessment algorithms discussed in scientific literature, as well as models commonly used in practice. It explores their strengths and weaknesses, implements the algorithms, and evaluates their predictive accuracy using historical data. For practical experiments, the algorithms (XGBoost, LightGBM, AdaBoost, Logistic Regression, and Random Forest) were selected based on their demonstrated effectiveness in credit risk assessment [5].

The paper is organized as follows. The introduction is presented in the first section. Section 2 provides a review of historical developments on the topic. Section 3 discusses modern credit risk assessment algorithms. Section 4 describes the data and empirical experiments. Finally, the paper concludes with a summary of key findings and remarks.

2 Historical developments survey

Historically, credit risk assessment has relied on statistical and descriptive methods, many of which continue to be used today.

One of the algorithms important to the history of credit risk assessment is the Merton model [19]. Building on the work of Black and Scholes [2] in option pricing, the model conceptualizes a company's equity as a European call option on its total enterprise value. Shareholders have a residual right to the company's assets and only get their money back when all debts are paid off. That is similar to a long position in a European call option: the shareholders have the right, but not the obligation, to "acquire" the company by repaying its debts. If the asset value exceeds the debt, they will repay the debts and retain the residual value. However, if the value of a company's assets does not exceed its debts, then similarly to not exercising the option, shareholders will default and allow the company to go insolvent.

From the creditor perspective, instead of a parallel to an European call option, the payoff instead resembles a long position in a risk-free, zero coupon bond, combined with a short position in a European put, as the model assumes full repayment without default risk at a single maturity date, whereas the short put adjustment reflects the possibility of losses.

Despite its influence, the Merton model has notable downsides and limitations, such as the assumption of debt maturing at a single terminal moment in time and constant interest rates, which is unrealistic.

Another one of the most commonly used statistical methods in credit risk assessment is logistic regression [13]. It determines the probability of an event based on a set of independent variables. There are three types of logistic regression: binary is when the response or a dependent variable is dichotomous; multinomial logistic regression allows the dependent variable to have three or more possible outcomes; and ordinal logistic regression is when the dependent variables have a sorted order.

Presently, there are three globally leading rating agencies: S&P Global, Moody's, and Fitch. Their ratings significantly impact the cost of borrowing for governments, municipalities, and corporations. The agencies predominantly rely on a combination of quantitative metrics, such as debt ratios, and qualitative judgment, such as review of default history, to assess credit risk.

3 Review of modern credit risk assessment algorithms

In this section, we describe the currently used modern algorithms for credit risk evaluation, examining their underlying principles. We also compare their similarities and differences. They may be best categorized into two groups: boosting algorithms and non-parametric algorithms.

3.1 Boosting algorithms

Many modern algorithms used in credit risk assessment belong to the category of boosting algorithms. In machine learning, boosting refers to an ensemble technique designed to reduce prediction errors by combining multiple weak learners into a strong one. Initially, all data samples are assigned equal weights. After each iteration, misclassified samples receive higher weights, prompting the algorithm to subsequently focus more on difficult cases. This iterative process continues until the overall prediction error is sufficiently minimized [9].

XGBoost

The XGBoost algorithm (eXtreme Gradient Boosting), introduced by Chen *et al.* [7], was first released in 2014. Known for its efficiency and scalability, it is widely used in risk modeling as a high-performance, multi-threaded boosting algorithm. The algorithm provides a framework for gradient boosting. This machine learning technique builds models sequentially, where each new model corrects the errors of the previous ones by minimizing a loss function using gradient descent. The framework is implemented in programming languages like C++ and Java. The goal of the algorithm is the same as that of logistic regression: to fulfill the forecast

$$\hat{y}_i = \sum_{j}^{d} w_j x_{ij},$$

where x_i is the data, which in financial modeling may be the interest coverage ratio, debt leverage or other indicators [21]. Note that $x_{ij} \in \mathbb{R}^{n \times d}$. Here i is the position of the element in the data set $X = \{x_1, x_2, \ldots, x_i, \ldots, x_n\}$, w_j is the weight coefficient and j is the position of the weight coefficient in the set $\theta = \{w_j \mid j = 1, \ldots, d\}$.

The model is trained by iteratively adjusting the parameters w_j to minimize a loss function that quantifies the error between the predictions $\hat{y}_i$ and the actual labels y_i, given the input data x_i. An objective function is defined for this purpose.

Previous experiments have shown that XGBoost is more efficient than traditional credit models [23]. The algorithm is scalable and can easily process very large data sets. The use of parallel computing and hardware optimization allows for a quick

training process. Nevertheless, XGBoost can be demanding regarding computer resources and sensitive to data noise or outliers.

Adaboost

Another gradient boosting–based machine learning algorithm used in credit risk assessment is AdaBoost, introduced by Freund and Schapire in 1995. The model creates a sequence of weak learners by using shallow trees. Using adaptive sample weights, every tree is trained on the entire dataset. Similarly to XGBoost, the trees are combined through weighted scoring, where better-performing trees are more influential in the final decision.

The algorithm has been applied to credit risk assessment. Zhang *et al.* [24] integrated the Long Short-Term Memory (LSTM) neural network (a deep learning approach) with the AdaBoost algorithm and evaluated its performance in practical scenarios. LTSM was designed by Hochreiter and Schmidhuber [11] to address limitations of the Recurrent Neural Network (RNN). RNNs use a hidden state to handle sequential data by capturing information from previous time steps. However, a challenge known as the vanishing gradient problem can occur, where gradients progressively diminish during training, making it difficult for the model to learn long-term dependencies. LTSM allows for information retention by using the forget, input, and output gates, which means a special memory unit to learn long-term data dependencies can be implemented. The forget gate removes information that is no longer necessary, as input at a certain time, x_t, and the output of the previous cell, h_{t-1}, are used and combined using a weight matrix, a bias, and an activation function:

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big).$$

Here

- W_f is the representation of the weight matrix associated with the forget gate;
- $[h_{t-1}, x_t]$ is the concatenation of previous hidden state and the present input;
- b_f is the forget gate bias;
- σ is the activation (sigmoid) function.

An output value of 0 indicates that the information should be forgotten, while a value of 1 signifies that it should be kept. The input gate determines which new information should be added to the cell state. It uses a sigmoid function to filter and regulate the incoming data, allowing only the most relevant information to pass through. A tanh function is used to generate a vector of candidate values, essentially representing potential information to be added to the cell state. The sigmoid output equation is

$$i_t = \sigma\big(W_i \cdot [h_{t-1}, x_t] + b_i\big),$$

whereas the candidate value generation formula is as follows

$$\widehat{C}_t = \tanh\big(W_c \cdot [h_{t-1}, x_t] + b_c\big).$$

The output gate controls which information from the current cell state $\widehat{C}_t$ is displayed as output and passed on to the next time step,

$$o_t = \sigma\big(W_o \cdot [h_{t-1}, x_t] + b_o\big).$$

Adaboost performs well when dealing with unbalanced datasets because of the adaptive adjustment of sample weights. Nevertheless, the algorithm has downsides like sensitivity to noisy data and outliers, which may lead to reduced performance if the dataset contains many such cases [4].

LightGBM

LightGBM, introduced by Ke *et al.* [15], is another gradient boosting decision tree algorithm similar to XGBoost and Adaboost. The authors note that efficiency and scalability in most other gradient boosting-based algorithms are lacking when the feature dimension and data size are large. The reason is that for all features, it is necessary to scan every feature and data instance to assess possible split points, which is time-inefficient. LightGBM implements gradient-boosting decision trees with two additional features: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GOSS excludes a sizeable proportion of data instances with small gradients and only uses the remainder to estimate information gain. GOSS can obtain quite an accurate estimation of the information gain with a much smaller data size, as data instances with larger gradients play a more important role in the computation of information gain. EFB bundles mutually exclusive features to reduce their count. Finding the optimal bundling of exclusive features is NP-hard, however a greedy algorithm offers a good approximation while reducing features with minimal accuracy loss.

GOSS prioritizes instances with large gradients and ignores smaller gradients, thus, reducing computational cost with minimal information loss. First, GOSS sorts data instances based on the absolute value of their gradients. Large gradients imply these instances are difficult for the model to predict, which is why they provide more information value. Top $a \times 100\%$ of the cases with the largest gradients are retained, where a is the large gradient data sampling ratio. From the remaining instances with smaller gradients, GOSS performs random sampling to select $b \times 100\%$ of the data, where b is the sampling ratio of small gradient data. To prevent the distortion of data distribution after sampling, a compensation factor of $(1-a)/b$ is introduced when calculating information gain, by which GOSS amplifies sampled data. Instances of small gradients are thus balanced in the information gain calculation despite the reduced sample size.

EFB combines features into a bundle while maintaining separability. Fields to keep track of the positions within the bundle (*binRanges*) and to store the cumulative size of bundled feature (*totalBin*) are initialized. For all features in the bundle, the number of bins is added to (*totalBin*), and it itself is appended to (*binRanges*). An empty feature *newBin* (size *numData*) is then calculated. For all data points, bundled features are checked, and if the feature value is non-zero, it is mapped into the combined feature space using *binRanges*. Ultimately, *newBin* is output as the merged feature and (*binRanges*) as the information about how the original features map into the combined feature.

LightGBM offers several advantages, such as support for parallel computation and GPU acceleration. Nevertheless, the model may overfit noisy data due to a large number of trees and underperform on smaller datasets as it uses leaf-wise splitting, which requires a sizeable amount of data to generalize well (cf. [16]).

3.2 Non-parametric algorithms

Nonparametric algorithms are machine learning methods that do not assume a fixed number of coefficients. Unlike parametric algorithms, which rely on a predetermined, fixed set of coefficients regardless of the dataset size, nonparametric methods allow the number of parameters to grow with the data. As a result, they make no assumptions about the underlying data distribution [14].

Random forest

Random forest is a machine learning algorithm developed by Breiman [3] in 2001. It aggregates the outputs of multiple decision trees to produce a single prediction. For classification tasks, the final output is determined by majority voting among the trees, while for regression tasks, the algorithm returns the average of their predictions. To increase model diversity and to reduce overfitting, two key component sub-algorithms of bagging and feature randomness are used.

To reduce variance in a noisy dataset, **b**ootstrap **agg**regat**ing** is introduced. This technique involves generating multiple training subsets by sampling with replacement from the original learning set $\mathcal{L} = \{(y_n, x_n),\, n = 1, \ldots, N\}$, where y_n represents the output (e.g., a class label), and x_n denotes the corresponding input features. A procedure (predictor) $\varphi(x, \mathcal{L})$ is used to predict the output y based on input x. Suppose there are multiple learning sets $\mathcal{L}_k$, where every learning set $\mathcal{L}_k$ is sampled from the same underlying data distribution as $\mathcal{L}$. The goal is predictor improvement, which is achieved by combining multiple predictor $\varphi(x, \mathcal{L}_k)$ outputs. If y is a number, then $\varphi_A(x) = E_{\mathcal{L}}[\varphi(x, \mathcal{L})]$, where $E_{\mathcal{L}}$ equals the expected value over different learning sets. Otherwise, if y is a class label, a majority vote is used - N_j is the number of predictors that predict class j, the class with the highest vote is then chosen: $\varphi_A(x) = \arg\max_j N_j$. When constructing each tree, Random Forest introduces feature randomness by selecting a random subset of features at each split rather than evaluating all available features. This reduces the tree correlation, enhances model diversity, and helps prevent overfitting.

The algorithm offers several advantages. While individual decision trees tend to fit the training data tightly, posing a risk of overfitting, Random Forest mitigates this issue by averaging the predictions of many uncorrelated trees, which reduces overall variance and improves generalization. Additionally, it is versatile and capable of handling both regression and classification tasks. However, this ensemble approach can be computationally intensive, as each tree must be independently constructed, making it less optimal in terms of time and resource efficiency.

K-nearest neighbors

Another machine learning algorithm used in credit risk assessment is the k-nearest neighbors (KNN) algorithm, a supervised method applicable to both classification and regression tasks. As a non-parametric approach, KNN makes no assumptions about the underlying data distribution. Instead, it operates on the principle that similar data points are likely to be located near each other in the feature space.

The algorithm selects a value for k, representing the number of nearest neighbors to consider. It then calculates the distance between the new data point and all

points in the dataset using a distance metric. The most common metric is Euclidean, which measures the straight-line distance between two points [12]. Other frequently used metrics include the Manhattan distance, which sums the absolute differences along each dimension, and the Hamming distance, used primarily for categorical variables. Onve the k closest neighbors are identified, the most common label among the neighbors is assigned, and the average of the neighbors' numerical values is taken.

Note that, while the KNN algorithm is simple to implement and requires no training (since it stores all data in memory), it can be computationally expensive, as distances must be recalculated for every new prediction.

Decision trees

Like KNN, the decision tree is a supervised, non-parametric machine learning algorithm used for both classification and regression. It features a hierarchical tree structure composed of a root node, internal nodes (branches), and leaf nodes.

A decision tree begins with a root node, which feeds into the internal nodes, to which the root node is an incoming branch. The root and decision nodes perform evaluations to create homogeneous terminal node subsets, representing every dataset outcome. Decision trees use a divide-and-conquer strategy by conducting a greedy search at each node to identify optimal local split points based on particular criteria. The process is repeated recursively in a top-down manner until at least a majority of records are classified under specific labels. Several decision tree algorithms exist. **Iterative Dichotomizer 3** (ID3), developed by Quinlan, is mainly used for classification by forming decision trees [22]. A randomly chosen subset of training data is used to build a tree. If all data is classified correctly, the process is halted – otherwise, misclassified examples are added to the subset, the process repeats.

The decision tree algorithms are advantageous because they require little data for preparation and visual representations make them easily interpretable, which is non-trivial in practice. Nevertheless, decision trees may overfit if pruning is not applied [20].

Support vector machines (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm introduced by Cortes and Vapnik in 1995 [8]. It is used for classification and regression tasks, and aims to find the optimal boundary, known as the hyperplane, for separating data into differing classes. The optimal hyperplane separates the classes with the largest possible distance between the hyperplane and the closest points from either class, called support vectors.

The algorithm has its advantages and drawbacks. It excels at handling non-linear data through the use of kernel tricks and achieves high accuracy, particularly when the number of features exceeds the number of samples. However, its interpretability is limited, restricting wider adoption in credit risk analysis [1]. Additionally, SVMs can be computationally expensive and are less suitable for larger datasets.

3.3 Comparison of the algorithms

The algorithms discussed previously share common principles behind their flow: they are all supervised techniques that allow for non-linear decision boundaries. Despite

these shared foundations, they also differ in terms of learning strategies and computational cost.

The boosting algorithms discussed in this work – XGBoost, Adaboost, and LightGBM – are all ensemble methods that combine predictions derived from several base learners to form a strong overall model. Furthermore, every one of these models builds trees sequentially, where each new tree aims to correct prior trees' errors. XGBoost and LightGBM use gradient boosting, where every new tree is trained to minimize the previous model's loss function using gradient descent. Conversely, AdaBoost is based on weight-based boosting, where sample weights are revised following each iteration. LightGBM, notably, introduces leaf-wise tree growth, which makes the model more prone to overfitting.

The non-parametric algorithms share less in common and operate somewhat differently – a decision tree, in essence, asks binary questions to split the data, random forests consist of multiple decision trees and combine the outputs for better accuracy, while KNN, as a lazy learning algorithm, aims to find closest data points to make a prediction. SVM draws a boundary (hyperplane) to separate classes by the widest possible margin.

The survey has shown that models with the highest computational cost are XGBoost, KNN, Random Forest, and SVM for prediction. Decision trees (if pruning is not used), random forest and LightGBM are the most prone to cases of overfitting. Boosting algorithms, in addition to SVM, are the most difficult to interpret. Next, we move on to the practical part of the research, where the following section presents empirical experiments conducted with the selected algorithms providing their performance on historical data.

4 Experiments on the dataset

To conduct experiments, the XGBoost, Adaboost, LightGBM, Random Forest, and Logistic Regression algorithms were chosen to be implemented owing to their experience-proven effectiveness.

4.1 Dataset description

The experiments utilized a dataset introduced by Liang *et al.* [18], comprising data on both solvent and bankrupt companies. The data includes financial information from the Taiwan Economic Journal covering the period from 1999 to 2009. The status of company bankruptcy was defined based on the Taiwan Stock Exchange business regulations. The authors applied two criteria for data selection: companies were required to have at least three consecutive years of complete financial records before the economic crisis, and there had to be a sufficient number of comparable companies of similar size within the same industry.

The dataset is provided as a single CSV file containing 6819 instances and 95 input features. Of these instances, 220 correspond to bankrupt companies, resulting in a highly imbalanced dataset. The classifier (whether a company is bankrupt or not) column was named 'Bankrupt?'.

This particular dataset was chosen because of its clear bankruptcy definition and, due to its unbalanced nature, providing a good reflection of the real-world challenges.

It also captures changes in time dynamics through variables such as Operating Profit Growth Rate. Furthermore, the dataset has a Kaggle score of 10/10 for completeness, source reliability, and usability, which is uncommon and reflects its high quality.

4.2 Tool implementation and metrics

A *Python* program has been developed to conduct experiments with the selected algorithms. The application outputs the defined evaluation metrics and can optionally plot precision-recall (PR) and receiver operating characteristic (ROC) curves with corresponding areas under the curve. XGBoost, LightGBM, Adaboost, Random Forest, and Logistic Regression algorithms were implemented using the *xgboost*, *lightgbm*, and *sklearn* libraries. The dataset was partitioned into training and testing sets, with 80% of the data used for training and the remainder reserved for guessing.

To investigate whether oversampling techniques could enhance the detection of bankrupt companies, the SMOTE algorithm was added. SMOTE synthetically generates minority group samples based on existing data, thereby increasing the representativeness of irregular data within the dataset [6]. Furthermore, the ADASYN [10] algorithm, which is similar to SMOTE but generates a different number of samples depending on an estimate of the local distribution of the class to be oversampled [17], was also added as a selectable option. Both algorithms were implemented using the *imblearn* library.

To assess the performance of the implemented models, the evaluation metrics used included precision, recall, F1-score, area under the precision-recall curve (AUC-PR), area under the receiver operating characteristic curve (AUC-ROC), and the Brier score.

4.3 Results and discussion

Following the implementation of the algorithms, all models and all combinations with oversampling algorithms were evaluated in the experiments. To ensure a fair comparison, a consistent set of hyperparameters was applied across all trials: $random\_state = 42$ for reproductibility, $n\_estimators = 200$ to control the number of trees for ensemble models, $learning\_rate = 0.2$, $max\_depth = 10$ to prevent overfitting, $subsample = 0.8$ for added randomness. For logistic regression, two additional hyperparameters were set as $max\_iter = 5000$ to ensure convergence in training, inverse of regularization strength $C = 10$ to allow more model flexibility. The experimental results are summarized in Table 1.

As presented in Table 1, LightGBM achieved the best precision (0.7222) in identifying bankrupt companies. However, its recall was low (0.2955), correctly detecting only 29.55% of actual bankruptcies. XGBoost had the highest F1-score, with slightly better recall (0.3409) but lower precision (0.5769). Regarding PR-AUC, which is particularly informative for imbalanced datasets, LightGBM again outperformed other models with a score of 0.5342. Logistic regression generally exhibited the poorest performance, likely due to linear assumptions. Overall, the results indicate that all models struggled with the minority class, underscoring the necessity for applying oversampling techniques.

Table 2 demonstrates that applying SMOTE substantially improved recall, albeit with a slight decrease in precision. Adaboost obtained the highest recall of 0.8409.

Table 1. Comparison of algorithms **without** oversampling,

No oversampling		XGBoost	Adaboost	LightGBM	Random forest	Logistic regression
Bankrupt?=1	Precision	0.5769	0.6000	0.7222	0.5000	0.3182
	Recall	0.3409	0.1364	0.2955	0.1364	0.1591
	F1-score	0.4286	0.2222	0.4194	0.2143	0.2121
Bankrupt?=0	Precision	0.9783	0.9719	0.9770	0.9719	0.9762
	Recall	0.9917	0.9970	0.9962	0.9955	0.9886
	F1-score	0.9850	0.9843	0.9865	0.9835	0.9805
	ROC-AUC	0.9470	0.9264	0.9490	0.9383	0.8467
	PR-AUC	0.4610	0.4001	0.5342	0.4717	0.2449
	Brier score	0.0235	0.0725	0.0245	0.0224	0.0307

Table 2. Comparison of algorithms **with** SMOTE oversampling

SMOTE		XGBoost	Adaboost	LightGBM	Random Forest	Logistic Regression
Bankrupt?=1	Precision	0.4510	0.2189	0.5238	0.3107	0.1939
	Recall	0.5227	0.8409	0.5000	0.7273	0.7273
	F1-score	0.4842	0.3474	0.5116	0.4354	0.3062
Bankrupt?=0	Precision	0.9840	0.9941	0.9834	0.9905	0.9900
	Recall	0.9788	0.9000	0.9848	0.9462	0.8992
	F1-score	0.9814	0.9447	0.9841	0.9678	0.9428
	ROC-AUC	0.9481	0.9310	0.9497	0.9423	0.8617
	PR-AUC	0.4989	0.4654	0.5403	0.4409	0.2884
	Brier score	0.0296	0.1161	0.0268	0.0456	0.0834

Table 3. Comparison of algorithms **with** ADASYN oversampling.

ADASYN		XGBoost	Adaboost	LightGBM	Random Forest	Logistic Regression
Bankrupt?=1	Precision	0.4583	0.2057	0.6000	0.3131	0.1928
	Recall	0.5000	0.8182	0.5455	0.7045	0.7273
	F1-score	0.4783	0.3288	0.5714	0.4336	0.3048
Bankrupt?=0	Precision	0.9833	0.9933	0.9849	0.9897	0.9900
	Recall	0.9803	0.8947	0.9879	0.9485	0.8985
	F1-score	0.9818	0.9414	0.9864	0.9687	0.9420
	ROC-AUC	0.9492	0.9344	0.9530	0.9434	0.8618
	PR-AUC	0.5055	0.4560	0.5575	0.4604	0.2905
	Brier score	0.0300	0.1184	0.0251	0.0450	0.0859

However, only 21.89% of all firms flagged as bankrupt were actually insolvent. LightGBM maintained the highest precision, though it declined from 0.7222 to 0.5238 after oversampling. PR-AUC scores also improved, with LightGBM achieving 0.5403. Furthermore, LightGBM produced the highest F1-score (0.5116). It is worth noting that Brier scores increased, which may be the cause of overfitting on synthetic samples.

Table 3 shows that ADASYN provides a similar effect to SMOTE by boosting recall at the expense of precision. Adaboost obtained the highest recall at 0.8182.

However, the precision score was relatively low at 0.2057. LightGBM delivered the highest precision at 60% and also produced the best F1-score (0.5714), surpassing the performance of all other models trained with SMOTE or without any oversampling technique.

5 Conclusions

The survey and experiments on modern machine learning algorithms for credit risk assessment lead to the following conclusions:

- The historical evolution of credit risk modeling reflects a shift from traditional statistical methods to advanced machine learning algorithms, largely enabled by the growing capacity to process and analyze large-scale data. Among the reviewed models, XGBoost, KNN, Random Forest, and SVM are identified as the most computationally demanding. Decision trees (without pruning), as well as Random Forest and LightGBM, demonstrate a greater tendency toward overfitting, particularly when applied to complex or imbalanced datasets. Moreover, boosting algorithms (along with SVM) pose significant challenges in interpretability, which may constrain their adoption in regulatory or decision-making contexts where model transparency is essential.
- The experimental results indicate that no single model consistently outperformed all others across all metrics. However, LightGBM, AdaBoost, and XGBoost emerged as the most effective overall. AdaBoost achieved the highest recall when combined with oversampling techniques (0.8409 with SMOTE and 0.8182 with ADASYN), while XGBoost and LightGBM produced the highest F1-scores, reflecting strong performance in balancing precision and recall.
- The experiments have shown that oversampling algorithms improve recall scores, albeit at the cost of precision. In this regard, ADASYN slightly outperformed SMOTE. Across all experimental configurations, the combination of LightGBM and ADASYN delivered the strongest performance in minority class identification, yielding the highest F1-score, ROC-AUC, and PR-AUC, as well as the lowest Brier score.

References

[1] D. Barbella, S. Benzaid, J. Christensen, B. Jackson, X. Qin, D. Musicant. Understanding support vector machine classifications via a recommender system-like approach. In *International Conference on Data Mining*, pp. 305–311, 2009. https://api.semanticscholar.org/CorpusID:16382246.

[2] F. Black, M. Scholes. The pricing of options and corporate liabilities. *J. Political Econ.*, **81**:637–654, 1973.

[3] L Breiman. Random forests. *Machine Learning*, **45**:5–32, 10 2001.

[4] J. Cao, S. Kwong, R. Wang. A noise-detection based AdaBoost algorithm for mislabeled data. *Pattern Recognit.*, **45**(12):4451–4465, November-December 2012.

[5] V. Chang, S. Sivakulasingam, H. Wang, S.T. Wong, M.A. Ganatra, J. Luo. Credit risk prediction using machine learning and deep learning: a study on credit card customers. *Risks*, **12**(11):174, 2024.

[6] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.*, **16**:321–357, 2002.

[7] T. Chen, C. Guestrin. XGBoost: a scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, San Francisco, CA, USA, 2016.

[8] C. Cortes, V. Vapnik. Support-vector networks. *Mach. Learn.*, **20**:273–297, 1995.

[9] M.A. Ganaie, M. Hu, A.K. Malik, M. Tanveer, P.N. Suganthan. Ensemble deep learning: a review. *Eng. Appl. Artif. Intell.*, **115**:105151, 2022.

[10] H. He, Y. Bai, E.A. Garcia, S. Li. Adasyn: adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1322–1328, 2008.

[11] S. Hochreiter, J. Schmidhuber. Long Short-Term Memory. *Neural Comput.*, **9**:1735–1780, 1997.

[12] IBM. What is the KNN algorithm? https://www.ibm.com/think/topics/knn/. Accessed 2025-03-19.

[13] IBM. What is Logistical regression?, 2023. https://www.ibm.com/think/topics/logistic-regression/. Accessed 2025-03-03.

[14] F. Imam, P. Musilek, M.Z. Reformat. Parametric and nonparametric machine learning techniques for increasing power system reliability: a review. *Information*, **15**(1):37, 2024.

[15] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T. Liu. LightGBM: a highly efficient gradient boosting decision tree. *Adv. Neural Inf. Process. Syst.*, **30**, 2017.

[16] A.A. Khan, O. Chaudhari, R. Chandra. A review of ensemble learning and data augmentation models for class imbalanced problems: combination, implementation and evaluation. *Expert Syst. Appl.*, **244**, 2024.

[17] G. Lemaître, F. Nogueira, C.K. Aridas. Imbalanced-learn: a Python Toolbox to tackle the curse of imbalanced datasets in machine learning. *J. Mach. Learn. Res.*, **18**(17):1–5, 2017.

[18] D. Liang, C.C. Lu, C.F. Tsai, G.A. Shih. Financial ratios and corporate governance indicators in bankruptcy prediction: a comprehensive study. *Eur. J. Oper. Res.*, **252**(2):561–572, 2016.

[19] R.C. Merton. On the pricing of corporate debt: the risk structure of interest rates. *J. Finance*, **29**:449–470, 1974.

[20] D.I. Mienye, N.R. Jere. A survey of decision trees: concepts, algorithms, and applications. *IEEE Access*, **PP**(99):1–1, 2024.

[21] D. Norkus. Kredito rizikos prognozavimas naudojantis mašininio mokymosi algoritmu „XGBoost“. Master's thesis, Vilniaus universitetras, 2020.

[22] J.R. Quinlan. Induction of decision trees. *Mach. Learn.*, **1**(1):81–106, 1986.

[23] S. Tao. XGBoost-based corporate failure risk prediction. *Wireless Internet Technology*, **15**:102–104, 2018.

[24] Q. Zhang, C. Zhang, X. Zhao. Credit risk classification prediction based on optimised adaboost algorithm with Long Short-Term Memory neural network (LSTM). *Adv. Econ. Manag. Pol. Sci.*, **87**:152–158, 2024.

REZIUMĖ

Aktualių kredito rizikos įvertinimo algoritmų palyginimas

S. Rimašauskas, I. Belovas, R. Gricius

Kredito rizikos įvertinimas yra būtinas priimant finansinius sprendimus, ypač investuojant į skolos vertybinius popierius. Kadangi obligacijų rinka yra didžiausia vertybinių popierių rinka pasaulyje, egzistuoja didelė paklausa įrankiams emitento kreditingumui įvertinti. Be to, informacija, nusakanti nemokumo tikimybę, yra naudojama ir kitose srityse, tokiose kaip rizikos valdymas. Mašininio mokymosi ir didžiųjų duomenų eroje atsirado būdų, leidžiančių automatizuotai įvertinti riziką remiantis dideliu kiekiu duomenų. Tradiciniai būdai kreditingumui įvertinti gali būti netikslūs, nes investuotojas gali būti šališkas arba klaidingai interpretuoti turimą informaciją. Moderniųjų įrankių, kurie leistų sumaniai apdoroti didelius informacijos kiekius, peržiūra bei palyginimas padės kuo objektyviau įvertinti emitento kredito riziką.

Raktiniai žodžiai: kredito rizika; mašininis mokymas; XGBoost; LightGBM; AdaBoost; logistinė regresija; atsitiktinis miškas