Proceedings of the 3rd International Conference on Applied Physics and Mathematical Modeling
DOI: 10.54254/2753-8818/2025.DL27968

Exploring the Trade-off Between Accuracy and Transparency in Credit Risk Prediction Models

Chuntian Ke

Department of Statistical Sciences, University of Toronto, Canada
Chuntian.ke@mail.utoronto.ca

Abstract. This study investigates four widely used models for credit risk prediction, Logistic Regression, Random Forest, XGBoost, and LightGBM, focusing on their ability to detect loan defaults under imbalanced and complex data conditions. Model performance was assessed using confusion matrices and key metrics including recall, precision and F-scores and analyzing the metric that best aligns with the purpose of lending institutions to evaluate a model's performance. This study analyzes the rationale for moving from Ordinary Least Squares (OLS) regression to logistic regression. The results indicate that while logistic regression provides transparency, it struggles with non-linear relationships and class imbalance. Random Forest, built on decision trees, improves stability but sacrifices interpretability. Two boosting methods, XGBoost and LightGBM, achieve superior predictive ability and efficiency with even lower transparency. Also introduces the evolution of decision trees into ensemble methods such as Random Forest, XGBoost, and LightGBM, and analyzes the structural differences of the decision trees employed within these models. Overall, the findings highlight the trade-off between ability to identify target customer of leading institutions and interpretability in credit risk modeling.

Keywords: Loan default, Logistic Regression, Random Forest, XGBoost, LightGBM

1. Introduction

In recent years, with the rapid advancement of financial technologies, banks and lending institutions have reached a new stage in credit risk assessment. Traditionally, credit risk evaluation heavily relied on experts' judgment, which was based on intuition and experience. Another method is using statistic models such as logistic regression, which assume linear relationships exist and require strict data conditions [1]. While these approaches have proven effective in many cases, limitations remain in handling complex, high-dimensional data. As the range of relevant features has expanded with the evolving social environment and the growing digitalization of financial services, these limitations have become more apparent.

The emergence of big data has introduced new challenges and opportunities to financial risk modeling. Financial institutions now have access to diverse large-scale datasets, which may not be well suited for traditional statistical models. This is due to the fact that such models are limited in capturing complex feature interactions and often fail to accurately represent the minority class under imbalanced data distributions [2]. However, borrower behavior, repayment dynamics, and market

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2753-8818/2025.DL27968

volatility frequently interact in nonlinear and high-dimensional ways, making these conventional models less effective in practice [2,3]. Machine Learning (ML) models are able to automatically detect such interactions and effectively fit highly nonlinear relationships, owing to their weaker assumption compared to traditional statistic model. Numerous empirical studies have shown that ML models, particularly random forests, consistently outperform traditional statistical models such as logistic regression across a wide range of credit risk applications [4].

Motivated by the increasing use and demonstrated advantages of machine learning in credit risk prediction, this paper aims to compare the performance and practical characteristics of four commonly used models: Logistic Regression, Random Forest, XGBoost, and LightGBM. Model performance is evaluated using the confusion matrix and standard classification metrics. The confusion matrix summarizes the counts of true and false predictions, while the metrics such as accuracy, recall, precision, and the area under the ROC curve (AUC) reflect the models' ability to correctly identify loan defaults. In addition, practical characteristics are considered to assess each model's applicability in real-world financial decision-making. These four models differ in complexity, learning type, and flexibility. By comparing their performance, we aim to understand which types of credit data each model is suited for.

This study focuses on evaluating each model's predictive ability and transparency, with particular attention to their performance in default probability prediction in credit risk tasks. Traditional statistical models such as Logistic Regression are easy to be interpret, since they provide parameter estimates that relate predictors to the outcome variable. By contrast, ML models often achieve higher predictive accuracy but are usually regarded as black box, making drivers behind the result are often difficult to interpret [5]. Transparent models allow financial institutions to explain lending decisions to applicants, while ML models help reduce lenders' risk by improving predictive outcomes. Therefore, in contexts where policy and regulation require fair lending practices, the trade-off between interpretability and accuracy becomes particularly important. As a result, conducting a cross-comparison between statistical models and machine learning approaches is both meaningful and necessary [4].

2. Model evaluation

2.1. Logistic regression

Logistic regression is a statistical model designed for binary outcomes, where the response takes values such as default or non-default [6]. It initially introduced in manufacturing and health related studies for analyzing dichotomous events [7]. It also adopted in the financial industry later and has become the baseline model for predicting loan default due to its transparency and interpretability. Ordinary least squares (OLS) regression serves as the foundation of regression analysis, but it is not appropriate when the dependent variable is binary. This is due to three main issues: predictions may fall outside the range of 0 to 1, the error cannot be assumed to follow a normal distribution, and the variance of the errors is not constant but instead depends on the predicted probability [7]. To address these limitations, logistic regression applies the logit transformation, which converts the linear predictor into a probability of default between 0 and 1. The logit transformation expresses the log-odds of default as a linear combination of borrower features. Formally, the model is given by:

$$logit(p) = ln(\frac{p}{1-p}) = \beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k \quad (1)$$

DOI: 10.54254/2753-8818/2025.DL27968

where p is the probability that a borrower defaults, and $X_1, X_{2,} \ldots, X_k$ represent borrower features such as income, age, or credit history [3]. Odds ratio, represented by $\exp(\beta_i)$, quantifies how the odds of the default change with each unit increase in X_i . By taking the natural logarithm of the odds ratio, we obtain the log-odds representation. Each coefficient β_i directly reflects the change in the log-odds of default associated with a one-unit increase in the corresponding predictor.

The coefficients of logistic regression are estimated using Maximum Likelihood Estimation (MLE) [8]. In credit risk prediction, this means finding the coefficients that make the observed default events occur with highest probability. the likelihood function can be written as:

$$L(\beta) = \prod_{i=1}^{n} p_i^{y_i} \tag{2}$$

where n is the sample size; $y_i \in \{0,1\}$, indicates whether the ith borrower defaults; p_i is the predicted probability of default given features of applications [7]. The logarithm is a monotonically increasing function and transforms the product form of the likelihood into a summation that is easier to optimize. For computational convenience, we maximize the log-likelihood instead of the likelihood function itself. The log-likelihood is defined as

$$l(\beta) = \sum_{i=1}^{n} [y_i ln(p_i) + (1 - y_i) ln(1 - p_i)] \tag{3}$$

Maximizing this function yields parameter estimates that best fit the observation.

In practical applications, the dataset is divided into a training set and a test set. The training set is used to fit the model, that is, to estimate the most appropriate parameters. However, the imbalance between default and non-default cases introduces significant bias during model training. Since non-defaulters usually represent the majority of cases in practice, the model may find that predicting all applicants as non-defaulters minimizes the overall classification error [9]. For example, suppose that in a dataset the prediction accuracy of a more advanced model is around 75%, while non-defaulters account for 80% of the cases. In this situation, simply predicting all applicants as non-defaulters would yield an accuracy of 80%, which yields a higher accuracy than the more advanced model. However, such inflated accuracy is not what lending institutions truly see. The financial loss from a single default typically far outweighs the profit from a non-defaulter. Lenders generally prefer to reject some potential non-defaulters rather than approve loans to applicants who are likely to default. They consider approving more non-defaulters only after ensuring that potential losses from defaulters are effectively controlled. In other words, what lending institutions truly aim for is to identify potential defaults accurately, rather than merely achieving a high accuracy.

When a trained logistic regression model is applied to the test set, its performance can be summarized using a confusion matrix, which consists of four components:

- True Positive (TP): correctly predicted defaulters.
- False Positive (FP): non-defaulters incorrectly predicted as defaulters.
- True Negative (TN): correctly predicted non-defaulters.
- False Negative (FN): defaulters incorrectly predicted as non-defaulters.

The overall accuracy metric is then defined as:

$$accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{4}$$

DOI: 10.54254/2753-8818/2025.DL27968

Since the denominator of the accuracy formula corresponds to the total number of cases in the test set, it remains constant. Under class imbalance, the contribution of TN far outweighs that of TP, making accuracy an inappropriate metric for evaluating the model's ability to identify defaulters. Instead, recall is often adopted as a more informative metric, as it directly measures the proportion of actual defaulters that are correctly identified by the model:

$$recall = \frac{TP}{TP+FN} \tag{5}$$

A high recall indicates that the model successfully captures most of the defaulters. Conversely, the metric that evaluates the model's ability to correctly identify non-defaulters, who can generate profit, is precision:

$$precision = \frac{TP}{TP+FP} \tag{6}$$

The closer the precision is to one, the fewer high-quality non-defaulters are misclassified. In practice, lending institutions place greater emphasis on recall to minimize the risk of granting loans to potential defaulters, while still considering precision to maintain a balance between risk control and business growth.

Since precision and recall measure different aspects of model performance, there is often a trade-off between them. To evaluate precision and recall together, the F-measure is commonly used. The general form is the weighted harmonic mean of precision and recall:

$$F_\beta = (1+\beta^2)\frac{precision \bullet recall}{\beta^2 \bullet precision + recall} \tag{7}$$

When β is set to 1, equal weight is assigned to precision and recall, and the resulting value is the F1-score. When β is set to 2, more weight is placed on recall relative to precision, and the resulting value is the F2-score. Consistent with the above discussion, the F2-score is more commonly used in credit risk prediction tasks.

2.2. Random forest

Decision trees are a wide applicable algorithm that can be applied to both classification and regression tasks [10]. The decision tree model is structured like a tree consisting of a root node, internal nodes, branches, and leaf nodes. At each internal node, the algorithm tests a feature and splits the data accordingly, with branches representing the possible outcomes of the test. The process continues recursively until a stopping condition is met, and the final decision is represented by a leaf node, which corresponds to a class label. In credit risk prediction, for example, a decision tree might first split applicants based on income. Applicants with income below a threshold are considered higher risk. Among those, it may further split based on age because younger applicants tend to have higher risk of default. Finally, it could test whether the applicant owns a house, with homeowners being more likely to be non-defaulters. Each path from the root to a leaf represents a series of decisions, and the leaf node gives the final prediction of default or non-default.

At each split, the algorithm automatically searches for the threshold that best separates the classes. The quality of a split is commonly measured by impurity reduction. One widely used criterion is the Gini impurity, defined as

DOI: 10.54254/2753-8818/2025.DL27968

$$Gini(S) = 1 - \sum_{i=1}^{n} p_i^2 \tag{8}$$

where p_i is the proportion of samples in node S that belong to class i[10]. For example, in a credit risk task, the model is performing binary classification. Assuming class 1 is defaulter and class 2 is non-defaulter. If node S only contain defaulters or non-defaulters, Gini value would be 0, meaning the node is completely pure. When evaluating a potential split, the decision tree does not only consider the purity of a single child node. Instead, it calculates the weighted average Gini impurity of all resulting nodes. A split is considered optimal when it produces the greatest decrease in Gini impurity, meaning the weighted average Gini value of the child nodes is much lower than that of the parent node. Once identified, this split becomes part of the final model, forming one of the internal decision rules of the tree. The optimal split method becomes part of the model's final structure, guiding how future applicants will be classified into defaulters or non-defaulters.

While decision trees are easy to interpret and can capture complex nonlinear relationships, they tend to overfitting. Overfitting is a phenomenon where models may generate overly complex rules that perfectly classify the training set but fail to generalize. If a decision tree is allowed to grow without constraints, it may generate very specific rules based on rare combinations of features. In credit risk tasks, rules coming from train set such as high income and home ownership imply lower risk are common and can generalize well to the test set. In contrast, rules like living in a particular apartment building and being aged between 30 and 40 implies high risk are rare, as this subgroup may contain only two or three individuals, perhaps just one in the training set. The default behavior of that single individual clearly cannot represent the entire group, and such rules are therefore unreliable and leading to overfitting.

Decision trees can apply several strategies to reduce overfitting, such as limiting the maximum depth, requiring a minimum number of samples per split or restricting the number of branches. While these constraints improve generalization, they also increase bias, meaning that the model becomes too simple to learn the true relationships in the data. In summary, a single unconstrained tree tends to overfit the training data, whereas a heavily constrained tree tends to underfit. To balance this trade-off, ensemble methods were developed. Random Forest is a combination of multiple decision trees and merges their outputs to make a final prediction, thereby reducing variance while maintaining relatively low bias [11].

It uses a technique called bagging (bootstrap aggregating), where each tree is trained on a bootstrap sample of the training data. This means that the training set for each tree is a different subset of the overall train set. Since the trees are trained on different data, they will learn different classification rules [12]. In addition to bagging, Random Forest introduces randomness in the selection of features. At each split, instead of evaluating all available features, the algorithm only considers a random subset of them. This further increases the diversity among trees, since different trees may rely on different features to form their classification rules. Very specific rules that appear in an overfitted tree are unlikely to be repeated across all trees. Random Forest determines the final classification through majority voting, so these spurious rules are effectively cancelled out and have no influence on the final classification. As a result, Random Forest is more stable and reliable than a single decision tree.

Random Forest also allows constraints to be applied to individual trees, such as limiting the maximum depth or requiring a minimum number of samples per split. These restrictions further reduce the risk of overfitting within each tree, while the ensemble structure ensures that the overall model still maintains strong predictive performance.

DOI: 10.54254/2753-8818/2025.DL27968

Through the explanation of its structure, Random Forest can be seen as a model that usually achieves a better performance than a single decision tree. However, this improvement comes with a cost: the model is more complex and requires higher computational resources, which makes it less efficient than a simple tree or logistic regression. In terms of interpretability, Random Forest is weaker than a single tree and logistic regression model since its prediction is the result of aggregating many trees, and it is difficult to trace back the exact decision path. Instead, we can still understand the model to some extent by looking at which features are most important for its decisions, or by checking how changing one feature affects the prediction. It is generally more interpretable than other completely "black box" ML models, such as boosting algorithms or deep learning methods.

2.3. XGboost

XGBoost (Extreme Gradient Boosting) is a method that builds numerous decision trees one after another. Each new tree focuses on the cases that the earlier trees predicted incorrectly. By adding trees in sequence, the model gradually improves its predictions [13]. Regularization is used in XGBoost to control the complexity of trees. It adds a penalty to the objective function when a tree becomes too deep or has too many leaves, which prevents the model from overfitting the training data [5]. It is designed to run faster and handle large datasets more efficiently. This is achieved through several structural optimizations: parallel computation, tree pruning and sparsity-aware processing [5].

In traditional decision trees or Random Forest, the search for the best split usually scans features and thresholds sequentially, meaning that each tree mainly relies on single-core computation, even though multiple trees can be trained in parallel. Parallel computation in XGBoost means that candidate split points for different features can be evaluated simultaneously, rather than sequentially. This allows the algorithm to take advantage of multi-core processors and significantly speed up the training process.

The tree pruning strategy in XGBoost allows trees to be fully grown first and then removes unnecessary branches in a post-pruning step if they do not improve the model's performance. In contrast, traditional decision trees (such as CART) often use pre-pruning, where the algorithm stops early if the improvement is not sufficient at each step. However, this may miss splits that only show their value in later stages. This approach avoids early stopping and reduces overfitting.

Sparsity-aware processing is the ability of model to automatically handle missing values and sparse features. Instead of requiring manual imputation, XGBoost learns a default direction for each split, so that samples with missing values are assigned to the branch that improves performance most. In contrast, traditional decision trees and Random Forests usually require preprocessing to fill in missing values before training.

These features make XGBoost not only more accurate in practice but also more practical for large-scale credit risk prediction tasks. Although it also generates a large number of trees, each tree is relatively shallow, as it is only used to correct the shortage of the previous ones. In contrast, the trees in Random Forest are complete and independent. This structural difference makes XGBoost more efficient, and its ability to use multi-core parallel computation further accelerates the fitting process. In credit risk prediction, XGBoost often achieves better performance than Random Forest, but it is more difficult to interpret because the final output results from the sequential combination of many trees.

DOI: 10.54254/2753-8818/2025.DL27968

2.4. LightGBM

Light Gradient Boosting Machine (LightGBM) is another gradient boosting framework, similar in principle to XGBoost but designed to be faster and more memory efficient. It achieves this through several key improvements over traditional gradient boosting implementations: histogram-based split finding,

Instead of evaluating every possible split point for continuous features, LightGBM groups feature values into discrete bins (histograms). Both XGBoost and LightGBM use histogram-based split finding to speed up the search for the best split. However, LightGBM further improves this approach by discretizing continuous features into fixed bins at the beginning of training [5]. While XGBoost needs to recompute histograms for both child nodes after each split, LightGBM applies a histogram difference technique, where one child's histogram can be obtained by subtracting the other from the parent. This reduces repeated calculations and further improves efficiency.

LightGBM uses a leaf-wise growth strategy, where at each step only the leaf with the largest gain is split [14]. This is different from both XGBoost and Random Forest, which grow trees level by level. In Random Forest, the level-wise growth is random by splitting on a random subset of features. In XGBoost, the level-wise growth is conducted with each new tree focusing on correcting the errors of the previous ones. By splitting only the leaf with the highest gain, LightGBM maximizes efficiency, leading to lower memory usage and faster fitting compared with XGBoost [14]. However, this strategy can also produce very unbalanced trees, where some branches grow very deep and capture highly specific conditions in the training data, which increases the risk of overfitting.

Sparse data are common in credit risk prediction, especially when categorical variables are transformed through one-hot encoding [14]. For example, a borrower's region may be represented by hundreds of dummy variables, only one of which takes the value 1 and others are all 0. Storing and processing these zeros explicitly wastes both memory and computation. LightGBM applies sparse-aware algorithms that automatically skip zero entries and only compute on non-zero values, making it far more efficient in handling such data. In addition, like XGBoost, LightGBM can automatically assign a default direction for missing values, which eliminates the need for manual imputation and further simplifies preprocessing.

LightGBM often achieves similar or better performance than XGBoost with faster training and lower memory cost. Compared with Random Forest, it is both more efficient and usually more accurate on imbalanced credit risk data. However, like XGBoost, LightGBM is a black-box model, and its interpretability remains limited compared to simpler models like Logistic Regression or Decision Trees.

3. Conclusion

This study has reviewed several machine learning models applied to credit risk prediction, including Logistic Regression, Decision Trees, Random Forest, XGBoost, and LightGBM. Each model has its own strengths and limitations. Logistic Regression offers interpretability, making it suitable as a baseline model, but it struggles with imbalanced data and non-linear relationships. Decision Trees provide intuitive rules but facing the issue of overfitting. Random Forest improves stability and performance by aggregating multiple trees at the cost of reduced interpretability. XGBoost and LightGBM, as boosting-based methods, generally achieve superior performance and efficiency on large-scale data, though their black-box nature raises challenges in interpretability.

DOI: 10.54254/2753-8818/2025.DL27968

In practice, the choice of model depends on the balance between ability to identify defaulter, efficiency, and interpretability required by lending institutions. While advanced boosting algorithms often achieve the highest predictive performance, simpler models may still be preferred when transparency and explainability are critical for regulatory compliance. Future research may focus on combining the predictive power of complex models with the interpretability of simpler approaches, helping financial institutions to identify the most suitable customers group while meeting regulatory requirements for explainability in loan decisions.

References

[1] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, “Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings, ” IEEE Transactions on Software Engineering, vol. 34, no.4, pp. 485–496, Jul. 2008, doi: https: //doi.org/10.1109/tse.2008.35

[2] T. R. Gatla, “Machine learning in credit risk assessment: Analyzing how machine learning models are transforming the assessment of credit risk for loans and credit cards, ” Journal of Emerging Technologies and Innovative Research (JETIR), vol. 10, no. 6, pp. 746–757, Jun. 2023, doi: 10.6084/m9.figshare.24044362

[3] K. K. Tiwari, R. Somani, and I. Mohammad, "Determinants of Loan Delinquency in Personal Loan, " International Journal of Management (IJM), vol. 11, no. 11, pp. 2566–2575, Nov. 2020, doi: 10.34218/IJM.11.11.2020.241.

[4] M. Moscatelli, F. Parlapiano, S. Narizzano, and G. Viggiano, “Corporate default forecasting with machine learning, ” Expert Systems with Applications, vol. 161, p. 113567, 2020, doi: 10.1016/j.eswa.2020.113567

[5] Z. Chen, H. Liu, and X. Huang, “Study on a prediction of P2P network loan default based on the machine learning LightGBM and XGboost algorithms according to different high dimensional data cleaning, ” Procedia Computer Science, vol. 131, pp. 973–980, 2018, doi: 10.1016/j.procs.2018.04.235.

[6] M. A. Sheikh, A. K. Goel, and T. Kumar, “An Approach for Prediction of Loan Approval using Machine Learning Algorithm, ” in Proceedings of the International Conference on Electronics and Sustainable Communication Systems (ICESC), IEEE, 2020, pp. 490–494, doi: 10.1109/ICESC48915.2020.CFP20V66-ART.

[7] L. M. Healy, “Logistic Regression: An Overview, ” Eastern Michigan University, College of Technology, Mar. 2006.

[8] I. P. Panagiotakos, “Introduction to multivariate regression analysis, ” Hippokratia, vol. 14, suppl. 1, pp. 23–28, 2010.

[9] A. Coser and C. Albu, “Predictive Models for Loan Default Risk Assessment, ” Economic Computation and Economic Cybernetics Studies and Research, vol. 53, no. 2, pp. 149–166, 2019, doi: 10.24818/18423264/53.2.19.09.

[10] J. R. Quinlan, “Induction of Decision Trees, ” Machine Learning, vol. 1, no. 1, pp. 81–106, 1986, doi: 10.1007/BF00116251.

[11] M. Madaan, A. Kumar, C. Keshri, R. Jain, and P. Nagrath, “Loan default prediction using decision trees and random forest: A comparative study, ” IOP Conf. Ser.: Mater. Sci. Eng., vol. 1022, p. 012042, 2021, doi: 10.1088/1757-899X/1022/1/012042.

[12] L. Breiman, "Bagging predictors, " Mahine Learning, vol. 24, no. 2, pp. 123–140, 1996, doi: 10.1007/BF00058655.

[13] J. H. Friedman, "Stochastic gradient boosting, " Computational Statistics & Data Analysis, vol. 38, no. 4, pp. 367–378, 2002, doi: 10.1016/S0167-9473(01)00065-2.

[14] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree, " in Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 2017, pp. 3149–3157.