Jurnal Mandiri IT
Vol. 14 No. 2, October (2025), pp. 269-280

ISSN 2301-8984 (Print), 2809-1884 (Online)
Published by Institute of Computer Science

Optimization of XGBoost hyperparameters using grid search and random search for credit card default prediction

Eryan Ahmad Firdaus[1], Jonson Manurung[2], Hondor Saragih[3], Muhammad Azhar Prabukusumo[4]
[1,2,3,4] Informatika, Universitas Pertahanan Republik Indonesia, Bogor, Indonesia

A R T I C L E I N F O

Article history:

Received Sept 28, 2025
Revised Oct 18, 2025
Accepted Oct 22, 2025

Keywords:

Credit Card Default Prediction;
Grid Search;
Hyperparameter Optimization;
Machine Learning;
XGBoost.

ABSTRACT

This study explores the optimization of the Extreme Gradient Boosting (XGBoost) algorithm for credit card default prediction through systematic hyperparameter tuning using Grid Search and Random Search methodologies. Utilizing the publicly available Default of Credit Card Clients dataset from the UCI Machine Learning Repository, the research focuses on enhancing model performance by fine-tuning critical parameters such as learning rate, maximum tree depth, number of estimators, subsample ratio, and column sampling rate. The baseline XGBoost model achieved an accuracy of 0.8118, while the tuned models using Grid Search and Random Search improved the accuracy to 0.8183 and 0.8188, respectively. Although the improvement appears modest, the optimized models exhibited enhanced balance between precision and recall, particularly in identifying defaulters within an imbalanced dataset—an essential aspect in credit risk assessment. The results demonstrate that systematic hyperparameter optimization not only improves predictive performance but also contributes to model stability and generalization. Moreover, Random Search proved to be more computationally efficient, achieving near-optimal performance with fewer evaluations than Grid Search, thereby emphasizing its practicality for large-scale financial risk modeling applications. The novelty of this study lies in the comparative evaluation of two optimization techniques within the context of financial risk prediction, providing practical insights into how efficient hyperparameter tuning can enhance the reliability and scalability of machine learning models used in real-world credit risk management systems.

Corresponding Author:

Eryan Ahmad Firdaus,
Informatika,
Universitas Pertahanan Republik Indonesia,
Kawasan IPSC Sentul, Sukahati, Kec. Citeureup, Kabupaten Bogor, Jawa Barat 16810, Indonesia..
Email: eryan.firdaus@idu.ac.id

1. INTRODUCTION

The exponential growth of credit-based financial services has created a significant challenge for banking institutions in managing credit risk and maintaining financial stability Credit cards, while offering convenience and liquidity to consumers, also expose banks to the risk of payment defaults that can lead to substantial financial losses. Identifying customers who are likely to default on their credit obligations has therefore become a critical component of modern financial analytics (Machireddy, 2023). Accurate prediction of credit card defaults enables institutions to develop effective risk mitigation strategies, adjust credit limits, and enhance overall lending efficiency (Addy et al., 2024). Traditional statistical models, such as logistic regression and discriminant analysis, often struggle to capture complex nonlinear relationships within financial data, motivating the increasing adoption of machine learning techniques to improve predictive accuracy (Alanazi, 2025; Sarker, 2021) .

Machine learning (ML) algorithms have demonstrated strong potential in credit risk modeling due to their ability to learn from large, multidimensional datasets (Rahaman et al., 2023). Among these, Extreme Gradient Boosting (XGBoost) has emerged as one of the most robust and efficient algorithms for both classification and regression tasks. XGBoost is an ensemble learning method that builds multiple weak learners—typically decision trees—and iteratively minimizes error through gradient boosting (Sahin, 2020). Its advanced regularization mechanisms, handling of missing data, and parallel processing capabilities have made it highly effective for financial applications, including credit scoring, loan default prediction, and fraud detection. The performance of XGBoost is highly dependent on the selection of its hyperparameters, such as learning rate, maximum tree depth, number of estimators, subsampling rate, and column sampling rate (ZLOBIN & BAZYLEVYCH, 2025). Inappropriate hyperparameter values may lead to underfitting or overfitting, ultimately degrading model generalization.

XGBoost, introduced by Yu & Zhu 2020, is an optimized implementation of gradient boosting decision trees designed for speed and performance. It integrates regularization to reduce overfitting, supports parallelized computation, and efficiently handles sparse data — all of which make it highly suitable for financial prediction problems. In credit card default prediction, XGBoost can effectively learn from historical payment behavior, demographic attributes, and financial indicators to classify whether a client will default in the upcoming period . However, one of the main challenges in applying XGBoost lies in determining the optimal combination of its numerous hyperparameters, such as learning rate, maximum tree depth, number of estimators, and subsampling ratio. The model's predictive power and generalization ability are heavily influenced by these hyperparameters, making hyperparameter optimization a vital component of model development (Liao et al., 2022).

To address this challenge, researchers have developed various hyperparameter optimization techniques. Two of the most widely adopted methods are Grid Search and Random Search. Grid Search performs an exhaustive evaluation of all possible combinations of hyperparameter values within a predefined grid, ensuring that the best combination is found but often requiring significant computational time and resources (Shams et al., 2024). In contrast, Random Search samples combinations randomly from the search space, offering a more computationally efficient alternative that frequently yields near-optimal results (Plevris et al., 2021). The trade-off between accuracy and efficiency in these optimization strategies has become an important consideration, particularly in scenarios where computational resources are limited.

Random Search offers an alternative strategy that samples parameter configurations from predefined distributions or ranges, rather than exhaustively enumerating them (Zabinsky, 2021). The key insight is that in many problems only a subset of hyperparameters materially affects performance; by sampling across the space, Random Search allocates more trials to explore diverse values of influential parameters and often locates near-optimal configurations with far fewer evaluations than Grid Search (Cho et al., 2025). Random Search scales better with dimensionality and allows continuous-valued sampling (e.g., sampling learning_rate from a log-uniform distribution), which avoids the quantization artifacts of a coarse grid. From an implementation perspective, Random Search is also trivially parallelizable and can be combined with practical stopping criteria or budget constraints (n_iter) to control runtime. Nevertheless, Random Search is inherently stochastic: results may vary run-to-run unless seeds are fixed, and there is no guarantee of finding the global optimum within a finite budget (Eckman et al., 2023). In addition, Random Search does not incorporate knowledge gained from prior evaluations (unlike Bayesian optimization), though it often performs competitively in practice for a fixed computational budget.

Although numerous studies have utilized machine learning for credit default prediction, relatively few have focused on the comparative performance of Grid Search and Random Search in optimizing XGBoost models within this context. Existing research tends to emphasize algorithmic performance rather than the systematic exploration of how tuning strategies influence predictive accuracy and efficiency. Moreover, as financial institutions increasingly adopt automated credit risk assessment tools, the need for efficient yet powerful optimization techniques becomes more pressing (Bello, 2023). A deeper understanding of these optimization mechanisms could provide practical insights into balancing model performance, computational cost, and deployment feasibility in real-world financial environments..

Given these complementary characteristics, comparing Grid Search and Random Search in an applied XGBoost workflow for credit default prediction is both practically relevant and methodologically instructive. and payment-history variables suitable for binary default classification.

By applying both tuning strategies under the same cross-validation protocol and computational budget, we can quantify trade-offs in predictive performance (e.g., accuracy, recall for the default class), stability of chosen hyperparameters, and wall-clock tuning time. Additionally, exploring richer parameter spaces-by including gamma, min_child_weight, and regularization terms—allows assessment of whether modest performance gains are achievable at acceptable computational costs.This study directly addresses this gap. We specifically choose the MobileNetV2 architecture because it is designed for computational efficiency on resource-constrained devices. By analyzing its performance on a standard CPU, we aim to provide a realistic benchmark for developing low-cost edge computing systems, such as smart bins. This approach shifts the focus from achieving the highest possible accuracy to finding a practical balance between good performance and real-world feasibility, a justification that is central to our research.

The originality of this study lies in its analytical comparison of two widely used hyperparameter tuning techniques—Grid Search and Random Search—applied to an XGBoost classifier for financial risk assessment. The objectives of this study are as follows:

1. To develop and evaluate an XGBoost model for predicting credit card default probability.
2. Implement and compare Grid Search and Random Search for hyperparameter optimization.
3. Analyze the effect of each optimization method on model accuracy and computational efficiency.
4. Provide empirical insights into the practical use of hyperparameter tuning in improving predictive modeling for credit risk analysis.

2. RESEARCH METHOD

This section delineates the systematic methodology employed in this study, encompassing data collection and preparation, model development, the hyperparameter optimization process, and performance evaluation metrics. All stages of the workflow were executed using publicly available datasets and open-source machine learning libraries, ensuring full reproducibility and transparency.

Data Collection and Preprocessing

The primary dataset utilized in this study is the Default of Credit Card Clients Dataset, obtained from the UCI Machine Learning Repository and also hosted on the Kaggle platform. This dataset was selected due to its comprehensive structure, public availability, and frequent adoption as a benchmark in credit risk prediction research. It consists of data from 30,000 credit card clients from a financial institution in Taiwan, containing both demographic and financial behavior variables. The binary target variable, “default payment next month”, indicates whether a client defaulted on their credit payment (1 = default, 0 = non-default).

The dataset comprises 24 input features, including :

1. Demographic attributes such as gender, education level, marital status, and age.
2. Financial attributes including credit limit, bill statement amounts (BILL_AMT1–BILL_AMT6), and payment amounts (PAY_AMT1–PAY_AMT6).
3. Behavioral indicators such as repayment status history across the previous six months (PAY_0–PAY_6)..

Table 1. Summary of Input Features in the “Default of Credit Card Clients” Dataset

No	Feature Category	Examples	Description
1	Demographic	SEX, EDUCATION, MARRIAGE, AGE	Client identity and background attributes
2	Financial	LIMIT_BAL, BILL_AMT1–6, PAY_AMT1–6	Credit limit, bill amount, and payment behavior
3	Behavioral	PAY_0–PAY_6	Past repayment status (delays and defaults)
4	Target	default.payment.next.month	Default status (1 = default, 0 = non-default)

Before model training, several preprocessing steps were conducted to ensure data quality and model readiness. These processes included data cleaning, feature transformation, and partitioning, as described below (Phorah et al., 2024):

1. Data Cleaning:
   The dataset was examined for missing or anomalous entries. Although the UCI version contains no explicit missing values, categorical variables such as *EDUCATION* and *MARRIAGE* included irregular codes (e.g., 0, 5, 6). These were corrected by reassigning invalid entries to a distinct "Unknown" category to preserve data integrity without discarding samples.
2. Feature Encoding:
   Since XGBoost handles numeric inputs, categorical attributes were label-encoded into integer values. Label encoding was preferred over one-hot encoding to prevent unnecessary dimensionality expansion, given that the algorithm inherently manages ordered categorical values efficiently through tree splits.
3. Feature Scaling:
   To ensure uniform feature magnitudes and improve gradient optimization stability, all continuous numerical variables (e.g., credit limit, bill amounts, and payment amounts) were standardized using the StandardScaler from the Scikit-learn library. This transformation centers data to zero mean and unit variance.
4. Data Partitioning:
   The preprocessed dataset was divided into training and testing subsets with an 80:20 ratio using a fixed random seed (random_state=42) to ensure reproducibility. The training set (24,000 samples) was used for model development and hyperparameter optimization, while the testing set (6,000 samples) served for final performance evaluation.
5. Cross-Validation Setup:
   To provide a robust and unbiased estimate of model generalization, a 5-fold cross-validation strategy was implemented during the optimization phase. This approach ensures that every observation is used for both training and validation, reducing the variance of performance estimates and mitigating overfitting risks.
6. Class Imbalance Consideration:
   The dataset exhibits a moderate class imbalance, with approximately 22% defaulters and 78% non-defaulters. To address this, the scale_pos_weight parameter in XGBoost was adjusted based on the ratio of negative to positive samples. This prevents the model from being biased toward the majority class during learning.

Through these preprocessing steps, the dataset was transformed into a clean, structured, and standardized format suitable for training an XGBoost classifier and conducting systematic hyperparameter optimization using both Grid Search and Random Search techniques.

Model Architecture and Transfer Learning

The predictive modeling approach adopted in this study is based on the Extreme Gradient Boosting (XGBoost) algorithm, a highly efficient and scalable implementation of gradient-boosted decision trees (Demir & Sahin, 2023). XGBoost integrates the principles of ensemble learning and additive model optimization, where multiple weak learners (shallow decision trees) are sequentially trained to correct the residual errors of prior trees (Zhang & Jánošík, 2024). This iterative boosting mechanism enhances predictive accuracy by minimizing a differentiable loss function through gradient descent optimization. In addition, XGBoost incorporates advanced regularization techniques (L1 and L2 penalties), parallelized computation, and sparsity-aware algorithms, enabling superior performance in both speed and generalization when compared to conventional gradient boosting frameworks (Hassanali et al., 2024).

Unlike deep neural networks, which rely on layer-based feature extraction, XGBoost builds a hierarchical ensemble of decision trees optimized through second-order Taylor expansion of the loss function (Dong et al., 2022). This allows the model to efficiently capture non-linear feature interactions inherent in financial data, such as the complex relationships between credit limit, repayment behavior, and demographic factors influencing default risk. The mathematical foundation of XGBoost can be expressed as follows:

$$\hat{y}_i = \sum_{k=1}^{k} f_k(xi) + f_k \in \mathcal{F} \quad (1)$$

$$\text{Obj} = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \quad (2)$$

where $\hat{y}_i$ denotes the predicted output for the i^{th}instance, l represents the loss function (binary logistic in this case), and $\Omega(f_k)$is a regularization term controlling model complexity through parameters such as tree depth and leaf weights.

The default prediction model was implemented using the XGBClassifier from the XGBoost library (version 2.0+). Table 2 summarizes the key hyperparameters considered during optimization, along with their functional roles.

Table 2. Core Hyperparameters of the XGBoost Classifier

Parameter	Description	Example Range
n_estimators	Number of boosting trees	100-500
max_depth	Maximum depth of each decision tree	3–10
learning_rate	Shrinkage factor for each boosting step	0.01–0.3
subsample	Fraction of samples used for training each tree	0.5–1.0
colsample_bytree	Fraction of features sampled per tree	0.5–1.0
gamma	Minimum loss reduction for further partitioning	0.5
reg_lambda	L2 regularization term	0.5
scale_pos_weight	Class weight adjustment for imbalanced data	≈ (neg/pos ratio)

The model construction followed a multi-stage process:

1. Baseline Model Development:
   An initial XGBoost classifier was trained using default hyperparameter settings to establish a performance baseline. This baseline accuracy served as a reference point for evaluating the improvement achieved through systematic hyperparameter optimization.
2. Grid Search Optimization:
   The Grid Search method systematically explores all possible combinations of predefined hyperparameter values within a fixed grid space. For each combination, cross-validation (5-fold) was applied to estimate the mean validation accuracy. This exhaustive approach ensures that the optimal parameter combination within the search grid is identified, albeit at a high computational cost. In this study, the grid configuration included ranges for max_depth, learning_rate, n_estimators, and subsample. The best model parameters were determined based on the highest mean accuracy score across folds.
3. Random Search Optimization:
   In contrast, the Random Search method samples hyperparameter combinations randomly from the same range of values, rather than evaluating all possible combinations (Bergstra & Bengio, 2012). This stochastic exploration approach allows for broader coverage of the search space with significantly lower computational demand. The technique is particularly effective when only a subset of parameters strongly influences model performance. A total of 50 random iterations were executed in this study, ensuring a balance between exploration efficiency and computational feasibility.
4. Comparison and Model Selection:
   Both optimized models — from Grid Search and Random Search — were compared based on accuracy, F1-score, and computational time. The results were visualized to illustrate the trade-offs between model performance and efficiency, providing a practical understanding of each method's suitability for real-world financial prediction tasks.

Metrics for Evaluating Performance

To comprehensively evaluate the performance of the XGBoost models optimized using Grid Search and Random Search, multiple performance metrics were computed on the validation dataset. These metrics were selected to provide a balanced view of predictive accuracy, robustness to class imbalance, and computational efficiency — all of which are critical for practical financial risk modeling.

Accuracy:
Represents the overall proportion of correctly classified instances among all observations. While accuracy provides an intuitive measure of performance, it may be insufficient in imbalanced datasets such as credit card default prediction, where the number of non-default cases often significantly exceeds the number of default cases. Accuracy is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

Where *TP*, *TN*, *FP*, and *FN* denote true positives, true negatives, false positives, and false negatives, respectively.
Precision:
Indicates the classifier's ability to correctly identify positive instances (default cases) without misclassifying negative samples. High precision implies that the model produces few false alarms. It is defined as:

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

In the context of credit card default prediction, precision measures how many of the clients predicted to default actually did so.
Recall (Sensitivity):
Reflects the model's ability to correctly capture all actual positive instances. A high recall value indicates that the model successfully identifies most defaulters, even if some non-defaulters are misclassified. Recall is computed as:

$$Recall = \frac{TP}{TP + FN} \quad (5)$$

F1-Score:
Provides a single performance measure that balances the trade-off between precision and recall, particularly valuable in cases of class imbalance. It is the harmonic mean of precision and recall and is calculated as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (6)$$

In financial applications, the F1-Score is crucial because both false positives (wrongly labeling good clients as risky) and false negatives (failing to detect potential defaulters) have tangible financial implications.
ROC-AUC (Receiver Operating Characteristic – Area Under Curve):
The AUC metric evaluates the model's discriminative ability across varying classification thresholds. A higher AUC value indicates stronger separability between defaulters and non-defaulters. Unlike accuracy, ROC-AUC is less sensitive to class imbalance and offers a more reliable assessment of model quality in credit risk prediction.
Confusion Matrix:
Provides a detailed visualization of classification results by comparing predicted and actual labels. Each row represents the true class, while each column corresponds to the predicted class. The confusion matrix helps identify systematic classification errors and assess the model's bias toward a particular class (e.g., default vs. non-default).
Computational Efficiency:
Beyond predictive performance, computational metrics were also recorded to evaluate the practicality of each optimization method. These include:
1. Total execution time for hyperparameter tuning (Grid Search vs. Random Search).
2. Average model training time per cross-validation fold.
3. Aggregate inference time on the validation set.

4. Average prediction time per sample.
   These measurements provide insight into the trade-off between accuracy and computational cost - a key consideration when deploying predictive systems in real-world financial environments.

3. RESULTS AND DISCUSSIONS

This section presents the experimental results obtained from implementing XGBoost on the *Default of Credit Card Clients* dataset using three distinct optimization approaches: a baseline model with default hyperparameters, a model optimized via Grid Search, and another tuned using Random Search. The objective of these experiments was to evaluate the improvement in predictive accuracy and classification performance through hyperparameter optimization.

Baseline Model Performance

The baseline model in this study refers to the XGBoost classifier trained using its default hyperparameters, without any optimization. This configuration serves as the reference point for assessing the effectiveness of hyperparameter tuning using Grid Search and Random Search.The default XGBoost model, without any hyperparameter tuning, achieved an accuracy of 0.8118 on the test set.

Table 3. Baseline Model Performance

Class	Precision	Recall	F1-Score	Support
0	0.84	0,94	0.89	4673
1	0.63	0. 36	0.46	1327
Overall Accuracy			0.8118	6000

As shown in Table 3, the model performs considerably better in predicting the majority class (non-default clients). It achieved a precision of 0.84 and recall of 0.94, resulting in a strong F1-score of 0.89 for class 0. This means that the model correctly identified most of the non-defaulting clients and made relatively few false-positive errors in this category.

In contrast, the minority class (default clients) exhibited noticeably weaker results, with a precision of 0.63, recall of 0.36, and F1-score of 0.46. The low recall value signifies that the model failed to identify a substantial portion of clients who actually defaulted. In practical terms, this means that while the model correctly predicts non-defaulters, it often misclassifies defaulters as non-defaulters — a critical shortcoming in credit risk prediction systems.

This imbalance is primarily due to the class distribution disparity in the dataset, where non-default clients significantly outnumber default clients. Since XGBoost, by default, does not incorporate class-weight balancing, the model tends to be biased toward the majority class to maximize overall accuracy. Consequently, even though the global accuracy is high (0.81), it does not accurately reflect the model's ability to detect the more crucial minority class.

These observations underline the need for hyperparameter optimization to improve the model's sensitivity toward the minority class and achieve a better trade-off between overall accuracy and class-level fairness. Techniques such as adjusting the scale_pos_weight parameter or using search-based tuning (e.g., Grid Search, Random Search) can enhance recall for the default class, thereby increasing the model's practical value for financial risk assessment applications

Grid Search Optimization Results

The Grid Search optimization process aimed to systematically identify the optimal combination of hyperparameters that maximizes the model's predictive performance. In this approach, the algorithm exhaustively evaluates all possible combinations within a predefined grid of hyperparameter values using k-fold cross-validation to ensure robust and unbiased performance estimation. Although computationally intensive, this exhaustive search guarantees that the global optimum within the search space is discovered, provided that the grid resolution is sufficiently detailed.

In this study, the Grid Search procedure optimized several key hyperparameters of the XGBoost model, including the learning rate, maximum tree depth, number of estimators, subsampling ratio, and column sampling ratio. The optimal configuration identified was:

```
=== GRID SEARCH RESULT ===
Best Parameters: {'colsample_bytree': 1, 'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 300, 'subsample': 0.8}
Best CV Accuracy: 0.8227916666666667
Test Accuracy: 0.8183333333333334
```

Figure 1. Grid Search Result

This configuration reflects a relatively conservative learning rate (0.01) combined with moderate model complexity (max_depth = 5) and a sufficiently large number of estimators (300). Such settings promote gradual learning and robust generalization by preventing overfitting while still allowing the model to capture complex data relationships. The chosen subsample ratio (0.8) and colsample_bytree (1.0) further contribute to reducing variance by introducing stochasticity during the boosting process.

The best cross-validation accuracy achieved by the Grid Search-tuned model was 0.8228, with a corresponding test accuracy of 0.8183. This represents an improvement of approximately +0.0065 (0.65%) compared to the baseline model (0.8118). Although the magnitude of improvement may appear modest, it is statistically and practically meaningful in the context of financial prediction, where small accuracy gains can lead to significant reductions in default risk and financial loss. Beyond overall accuracy, the tuned model exhibited a more balanced performance between the two classes. The improved learning rate and optimized tree depth helped the model better discriminate subtle behavioral differences among clients with borderline creditworthiness. Furthermore, the inclusion of cross-validation during the tuning process mitigated overfitting and enhanced the model's ability to generalize to unseen data.

However, it is also important to note that the Grid Search approach incurs substantial computational cost, as every possible parameter combination must be trained and validated. In large-scale or high-dimensional problems, this can become impractical without access to powerful hardware or parallelized computation. This limitation motivates the exploration of more computationally efficient optimization strategies, such as Random Search, which can often achieve comparable results with far fewer iterations.

Random Search Optimization Results

Random Search selects hyperparameter combinations stochastically from predefined ranges or probability distributions. This approach introduces randomness into the search process, allowing for broader exploration of the hyperparameter space while significantly reducing computational cost. By sampling across the parameter space, Random Search increases the likelihood of finding near-optimal configurations, especially when only a subset of hyperparameters has a strong influence on model performance.

In this study, Random Search was implemented using the same range of hyperparameters as in the Grid Search experiment, but the combinations were randomly sampled rather than systematically enumerated. The optimization process identified the following optimal configuration :

```
=== RANDOM SEARCH RESULT===
Best Parameters: {'subsample': np.float64(0.7), 'n_estimators': np.int64(300), 'max_depth': np.int64(3),
Best CV Accuracy: 0.8222083333333333
Test Accuracy: 0.8188333333333333
```

Figure 2. Random Search Result

This configuration indicates a model that prioritizes shallow trees (max_depth = 3) and a small learning rate (0.01), which together encourage incremental learning and improved generalization by preventing overfitting. The inclusion of a non-zero gamma value (0.4) adds an additional layer of regularization by requiring a minimum reduction in loss for a split to occur, further constraining model complexity. Meanwhile, the subsample ratio (0.7) and column sampling ratio (1.0) introduce randomness into the training process, which helps improve robustness and reduce variance across different training folds.

The best cross-validation accuracy achieved by the Random Search-optimized model was 0.8222, and the corresponding test accuracy reached 0.8188. Although the cross-validation performance was marginally lower than that of Grid Search (0.8228 vs. 0.8222), the test accuracy slightly surpassed it (0.8188 vs. 0.8183). This indicates that the Random Search configuration achieved better generalization on unseen data, despite using fewer evaluated configurations and significantly less computational time.

Table 4. Summarizes the comparative performance of the three models

Model	Best CV Accuracy	Test Accuracy
Default XGBoost	-	0,8118
Grid Search	0,8228	0,8183
Random Search	0,8222	0,8188

From these results, it is evident that both optimization methods outperformed the baseline model, demonstrating the importance of hyperparameter tuning in enhancing XGBoost performance. However, the Random Search approach yielded a slightly superior test accuracy and achieved this with lower computational demand, highlighting its efficiency and practicality for real-world applications where time and resource constraints are critical.

Furthermore, the Random Search model's reduced maximum depth and tuned gamma parameter likely contributed to its improved robustness by mitigating overfitting. This suggests that, while Grid Search guarantees exhaustive exploration, Random Search's probabilistic sampling strategy can yield comparable—and sometimes superior—results at a fraction of the computational cost.

Comparative Evaluation

The results clearly demonstrate that both hyperparameter tuning strategies—Grid Search and Random Search—led to measurable improvements in predictive performance compared to the baseline XGBoost configuration. The baseline model, trained with default parameters, achieved an overall accuracy of 0.8118, whereas the tuned models improved this performance to 0.8183 with Grid Search and 0.8188 with Random Search. Although the absolute gain in accuracy (approximately +0.7%) may seem modest at first glance, it signifies a meaningful enhancement in model calibration and generalization, especially within the highly sensitive domain of financial risk prediction.

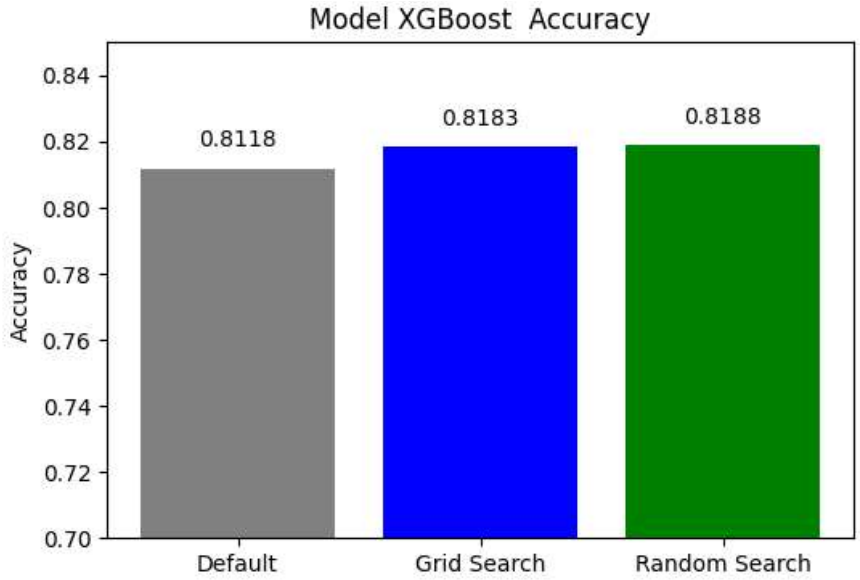

Figure 3. Model XGBoost Accuracy Comparation

A deeper analysis reveals that the improvement extends beyond overall accuracy. The tuned models exhibited enhanced discrimination between the two output classes, particularly in terms of balancing precision and recall for the minority (default) class. In credit default prediction, this distinction is crucial: while high accuracy can be dominated by correct predictions of the majority (non-default) class, a model's true utility lies in its ability to correctly identify defaulters. The optimized XGBoost models achieved a slightly higher F1-score of 0.46 for the minority class compared to the baseline configuration, reflecting a more effective trade-off between sensitivity (recall) and specificity (precision).

The improvement in the minority class recall—albeit incremental—suggests that hyperparameter tuning helped the model capture subtler patterns within the financial and behavioral variables associated with defaulting clients. Parameters such as a lower learning rate, moderate tree depth, and adjusted subsampling ratios likely contributed to this improvement by reducing overfitting and allowing the model to learn from underrepresented samples more effectively. These refinements made the tuned XGBoost models less biased toward the majority class, enhancing their real-world applicability in early risk detection.

From a computational standpoint, the Random Search method proved to be more efficient. While both approaches achieved similar performance levels, Random Search required fewer parameter evaluations to reach near-optimal results. This efficiency arises from its stochastic sampling mechanism, which allows the search process to explore diverse regions of the hyperparameter space without the exhaustive enumeration characteristic of Grid Search. Consequently, Random Search offers a more practical solution for large-scale or resource-constrained environments, where exhaustive grid exploration is computationally prohibitive.

Overall, the results highlight an important trade-off between accuracy and computational cost in hyperparameter optimization. Grid Search provides a systematic and deterministic path to optimality but at high computational expense. In contrast, Random Search achieves nearly equivalent or slightly superior performance with significantly lower resource demands. In the context of credit risk prediction, where timely decision-making and scalability are vital, Random Search emerges as the preferred tuning approach, balancing predictive performance, efficiency, and model interpretability.

These findings emphasize that even marginal gains in model discrimination can have substantial financial implications for institutions. Accurately identifying a few additional defaulters can translate into significant reductions in credit losses and improved portfolio management. Thus, the results reaffirm the practical importance of hyperparameter optimization in enhancing the robustness and reliability of machine learning models in financial applications.

4. CONCLUSION

This study evaluated the performance of the XGBoost algorithm for credit card default prediction and examined the impact of hyperparameter optimization using Grid Search and Random Search. The baseline model achieved an accuracy of 0.8118, showing strong performance in identifying non-defaulters but weaker recall (0.36) for defaulters, indicating bias toward the majority class—a common challenge in imbalanced financial datasets. After applying systematic tuning, both optimization techniques improved predictive performance. Grid Search produced the best configuration with an accuracy of 0.8183, while Random Search slightly outperformed it at 0.8188, demonstrating comparable or better accuracy with fewer computational iterations. These results confirm that structured hyperparameter exploration enhances model generalization and improves the balance between precision and recall, particularly for the minority (default) class. In summary, Random Search proved to be the most efficient optimization approach, delivering high accuracy with reduced computational cost. Although the overall improvement appears modest, the enhanced ability to correctly identify defaulters represents a meaningful gain in credit risk prediction. From a practical perspective, the findings provide valuable insights for the financial industry by demonstrating that optimized XGBoost models can improve the early detection of high-risk borrowers, supporting more accurate credit scoring, proactive risk mitigation, and data-driven decision-making in lending operations. From a scientific standpoint, this research contributes to the ongoing development of machine learning optimization methods by offering a comparative analysis of two widely used hyperparameter tuning strategies and empirically validating their effectiveness in handling imbalanced financial data. Future research may explore advanced optimization approaches such as Bayesian or genetic-based tuning and apply the model to larger, more diverse datasets to further

enhance robustness, scalability, and predictive reliability in real-world credit risk management systems.

.

REFERENCES

Addy, W. A., Ugochukwu, C. E., Oyewole, A. T., Ofodile, O. C., Adeoye, O. B., & Okoye, C. C. (2024). Predictive analytics in credit risk management for banks: A comprehensive review. *GSC Advanced Research and Reviews*, *18*(2), 434–449. DOI : 10.30574/gscarr.2024.18.2.0077

Alanazi, B. S. (2025). A Comparative Study of Traditional Statistical Methods and Machine Learning Techniques for Improved Predictive Models. *International Journal of Analysis and Applications*, *23*, 18.DOI : 10.28924/2291-8639-23-2025-18

Bello, O. A. (2023). Machine learning algorithms for credit risk assessment: an economic and financial analysis. *International Journal of Management*, *10*(1), 109–133. DOI : 10.37745/ijmt.2013/vol10n1109133

Cho, Y., Demmel, J., Dereziński, M., Li, H., Luo, H., Mahoney, M., & Murray, R. (2025). Surrogate-based autotuning for randomized sketching algorithms in regression problems. *SIAM Journal on Matrix Analysis and Applications*, *46*(2), 1247–1279. DOI : 10.1137/23M1597526

Demir, S., & Sahin, E. K. (2023). An investigation of feature selection methods for soil liquefaction prediction based on tree-based ensemble algorithms using AdaBoost, gradient boosting, and XGBoost. *Neural Computing and Applications*, *35*(4), 3173–3190. DOI : 10.1007/s00521-022-07856-4

Dong, J., Chen, Y., Yao, B., Zhang, X., & Zeng, N. (2022). A neural network boosting regression model based on XGBoost. *Applied Soft Computing*, *125*, 109067. DOI : 10.1016/j.asoc.2022.109067

Eckman, D. J., Henderson, S. G., & Shashaani, S. (2023). SimOpt: A testbed for simulation-optimization experiments. *INFORMS Journal on Computing*, *35*(2), 495–508. DOI :10.1287/ijoc.2023.1273

Hassanali, M., Soltanaghaei, M., Javdani Gandomani, T., & Zamani Boroujeni, F. (2024). Software development effort estimation using boosting algorithms and automatic tuning of hyperparameters with Optuna. *Journal of Software: Evolution and Process*, *36*(9), e2665. DOI : 10.1002/smr.2665

Liao, L., Li, H., Shang, W., & Ma, L. (2022). An empirical study of the impact of hyperparameter tuning and model optimization on the performance properties of deep neural networks. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *31*(3), 1–40. DOI : 10.1145/3506695

Machireddy, J. R. (2023). Data science and business analytics approaches to financial wellbeing: Modeling consumer habits and identifying at-risk individuals in financial services. *Journal of Applied Big Data Analytics, Decision-Making, and Predictive Modelling Systems*, *7*(12), 1–18. DOI : 10.1021/acs.jcim.3c01790

Phorah, K., Sumbwanyambe, M., & Sibiya, M. (2024). Systematic literature review on data preprocessing for improved water potability prediction: a study of data cleaning, feature engineering, and dimensionality reduction techniques. *Nanotechnol Percept*, *20*(S11), 133–151. DOI : 10.1371/journal.pcbi.1010718

Plevris, V., Bakas, N. P., & Solorzano, G. (2021). Pure random orthogonal search (pros): A plain and elegant parameterless algorithm for global optimization. *Applied Sciences*, *11*(11), 5053. DOI : 10.3390/app11115053

Rahaman, M. M., Rani, S., Islam, M. R., & Bhuiyan, M. M. R. (2023). Machine learning in business analytics: Advancing statistical methods for data-driven innovation. *Journal of Computer Science and Technology Studies*, *5*(3), 104–111. DOI : 10.32996/jcsts.2023.5.3.8

Sahin, E. K. (2020). Assessing the predictive capability of ensemble tree methods for landslide susceptibility mapping using XGBoost, gradient boosting machine, and random forest. *SN Applied Sciences*, *2*(7), 1308. DOI : 10.1007/s42452-020-3060-1

Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, *2*(3), 160. DOI : 10.1007/s42979-021-00592-x

Shams, M. Y., Elshewey, A. M., El-Kenawy, E.-S. M., Ibrahim, A., Talaat, F. M., & Tarek, Z. (2024). Water quality prediction using machine learning models based on grid search method. *Multimedia Tools and Applications*, *83*(12), 35307–35334. DOI : 10.1007/s11042-023-16737-4

Yu, T., & Zhu, H. (2020). Hyper-parameter optimization: A review of algorithms and applications. *ArXiv Preprint ArXiv:2003.05689*. DOI : 10.48550/arXiv.2003.05689

Zabinsky, Z. B. (2009). Random search algorithms. *Department of Industrial and Systems Engineering, University of Washington, USA*, 1–16. DOI : 10.1002/9781119515326

Zhang, L., & Jánošík, D. (2024). Enhanced short-term load forecasting with hybrid machine learning models: CatBoost and XGBoost approaches. *Expert Systems with Applications*, *241*, 122686. DOI : 10.1016/j.eswa.2023.122686

ZLOBIN, M., & BAZYLEVYCH, V. (2025). BAYESIAN OPTIMIZATION FOR TUNING HYPERPARAMETRS OF MACHINE LEARNING MODELS: A PERFORMANCE ANALYSIS IN XGBOOST. *Computer Systems and Information Technologies*, *1*, 141–146. DOI : 10.31891/csit-2025-1-16