Original article

Journal of High School Science

Advanced loan default prediction models using Machine Learning boosting algorithms

Chi Ton Cong Nguyen

Submitted: August 26, 2025, Revised: version 1, August 28, 2025, version 2, August 30, 2025
Accepted: August 31, 2025

Abstract

This research presents an advanced scientific approach using machine learning (ML) models with boosting algorithms and a data-driven modeling approach to achieve ~ 98% prediction accuracy for credit risk evaluation. The study was conducted using a public, loan-level dataset from Freddie Mac for the post-2020 period, and identified multiple credit risk factors that influenced the likelihood of loan default. The research examined whether ML boosting algorithms, including Gradient Boosting, XGBoost, and Light GBM, outperformed Logistic Regression in predictive performance. The paper proposes novel ML-based credit risk algorithms to address challenges, including data imbalance, hyperparameter optimization, and robust cross-validation, to achieve reliable estimation. For comprehensiveness and robustness, model performance was evaluated using a suite of key metrics, including accuracy, sensitivity, specificity, true and false positive rates, AUC, F1 scores, and ROC analysis. The empirical results of the paper demonstrated that ensemble methods consistently achieved superior accuracy compared to single-model approaches. The paper found that XGBoost and Light GBM were the top performers with 98% accuracy after optimization and 5-fold cross-validation. The findings demonstrated that ML models using boosting algorithms, especially XGBoost and Light GBM, achieved remarkable accuracy in distinguishing between “good” and “bad” loans compared to the traditional logit model, without exhibiting signs of overfitting. By outperforming current models for predicting loan defaults, the result carries significant implications for lenders, regulators, and policymakers in the financial industry, providing more robust tools for credit risk modeling, facilitating the development of FinTech-driven lending solutions, and supporting the preservation of financial stability.

Keywords

Machine Learning, Predictive modeling, Gradient Boosting, Light Gradient Boosting Machine (LightGBM), Extreme Gradient Boosting (XGBoost), Artificial Intelligence, Credit risk, Mortgage loan defaults, Statistical models

---

Chi Ton C. Nguyen, James Madison High School, 2500 James Madison Dr, Vienna, VA 22181, USA.
chiton0408@gmail.com

https://doi.org/10.64336/001c.144823

1 Introduction

One of the key tasks in risk management for the financial sector is to achieve sound credit risk assessment. In lending practice, credit scoring models are primary tools that financial institutions use to assess the creditworthiness of borrowers, estimate the likelihood of borrower default, manage lending risks effectively, and make prudent investments (1). Among different types of loans, mortgage loans pose a particularly significant risk due to their large values and long terms for repayment. As a result, to mitigate credit risks and resulting losses due to mortgage loan defaults, it is essential for banks and financial institutions to develop accurate predictive models.

Understanding and accurately predicting mortgage loan defaults is essential not only for minimizing credit losses among lenders but also for promoting broader macroeconomic stability. Rising default rates, especially widespread subprime mortgage defaults, can lead to a series of financial disruptions, as demonstrated during the 2008 global financial crisis (2). Advancing credit scoring models remains a critical focus of both academic research and financial risk management in the financial sector.

In the United States, the increased availability of consumer financial data and the need for increased risk management systems have led to the evolution of credit scoring systems from traditional statistical methods, which have been widely adopted due to their ease of interpretation and regulatory acceptance (3), to artificial intelligence (AI) or machine learning (ML) credit scoring models (4, 5). The use of statistical and AI/ML-based credit scoring systems has grown rapidly in response to the increasing availability of consumer financial data and the need for more robust risk. ML models can handle a large data of borrower behavior, loan characteristics, and risk profiles efficiently, and learn different patterns and relationships in the data to build more accurate models than traditional credit scoring models (4).

Given the dynamic nature of the economy and more complex borrowers' behavior, it is essential for risk assessment models to remain adaptive, continually updating their algorithms to reflect new data and emerging insights. It is necessary to evolve from standard credit scoring procedures to more sophisticated, data-driven methodologies that incorporate ML and advanced predictive analytics.

To achieve the goal of developing highly predictive models for credit scoring assessment, this research collected Freddie Mac loan-level data with more than 100,000 loans, including a rich set of characteristics of borrowers, loan records, macroeconomic factors, and property factors that could have a potential impact on the repayment of mortgage loans. Recent advancements in ML have introduced more flexible and potentially more accurate credit scoring models, such as gradient boosting and ensemble methods, which can capture complex nonlinear relationships in the data (4, 5). Hence, this paper explored ML boosting algorithms to enhance the prediction accuracy for mortgage default models, and to create efficient early warning systems to reduce credit risks and losses. Having accurate mortgage default models is crucial because it helps mortgage lenders remain competitive by managing risk efficiently and pricing the portfolios effectively. It also helps regulators monitor financial stability, make timely and effective decisions or interventions, and create sound lending practices in the finance sector.

To help lenders, financial investors, and professionals make proactively well-informed decisions; this research presents a novel approach to successfully predict loan defaults in the U.S. housing market. The main objective of this analysis was to examine if ML boosting algorithms could be used to predict loan defaults. The key hypothesis was that ML boosting algorithms could offer higher prediction accuracy than traditional Logistic Regression [logit] in predicting mortgage loan defaults. The results of the paper show that all the boosting algorithms showed improvement in prediction compared to the Logistic Regression. The study found that ML models using boosting algorithms, especially XGBoost and Light GBM, demonstrated superior performance compared to the traditional logit model in distinguishing between "good" and "bad" loans.

The outline of the rest of the paper includes five sections. Section 2 presents data overview, data preprocessing, feature selection, extraction, and correlation analysis. Section 3 discusses ML methodology and presents the three ML boosting algorithms, including Gradient Boosting, XGBoost, and Light GBM, in comparison with Logistic Regression. Section 4 discusses how the ML models were constructed, optimized with hyper-parameter tuning and evaluated. Section 5 shows the results and the discussion of the results. The last section concludes the study.

2 Data

2.1 Data overview

To achieve the goal of developing accurate mortgage loan default models, the study employed a large dataset of mortgage loans originated in 2020, with loan performance observed up to September 2023. The data consisted of more than 100,000 loans, which were mostly 30-year fixed-rate mortgages and extracted from Freddie Mac's public loan-level data (6). The dependent variable was Mortgage loan default status, which was defined as 1 if the mortgage loan defaulted with missing payments for at least 6 months, and 0 otherwise. Mortgage loan delinquencies were cleaned and added to the loan-level dataset. After cleaning the data, the data contained ~ 10.5% loan defaults (N = 10,867), and the rest being loans that were repaid on schedule or paid in full or missing payments for less than 6 months (N = 92,732). The data consisted of 18 independent variables, including loan, borrower, and property characteristics. Other features, such as macroeconomic data including unemployment rates and inflation rates, were also added to the data based on geographical location and individual loan performance.

2.2 Data preprocessing

To mitigate the challenge of class imbalance between the default and non-default loans (10.5% vs. 89.5%), a stratified data partition was used. To further balance the dataset and ensure equal representation of "good" and "bad" loans in the training and testing datasets, the Synthetic Minority Oversampling Technique (SMOTE) was applied to synthetically increase the minority class, in combination with under-sampling the majority class. Specifically, for each minority instance, SMOTE identifies its k nearest neighbors, randomly selects one neighbor to gather a synthetic sample; this process is repeated until the minority class reaches the desired level of representation relative to the majority class (7).

SMOTE addresses class imbalance by creating synthetic minority samples using an interpolation approach, which helps mitigate overfitting that can result from merely duplicating minority observations. As a result of applying SMOTE to increase the minority class and under-sampling the majority class, the final balanced dataset of more than 65,000 loans achieved an equal distribution of 50% loan defaults and 50% non-default loans across both training and testing data.

To fill in missing values, only for the Current Unpaid Balance (CUPB) and the Current Loan-to-Value (CLTV), the mean imputation process was implemented. However, the missing rate was negligible at 0.06%. After data preprocessing was completed, variable constructions, feature selection, and exploratory data analysis of mortgage loans were performed. Feature selection was used to identify relevant variables for model construction. Correlation analysis and recursive feature extraction were performed simultaneously to search for features that significantly improved the predictions of mortgage loan defaults. The data was cleaned, balanced, preprocessed, and analyzed using R and Python.

2.3 Feature Extraction

Freddie Mac data was used in this paper to predict mortgage loan defaults because it has a rich set of features of borrower, loan, and property characteristics that are consistent with previous work related to credit risk assessment (8). The data has a comprehensive set of features, including Credit score (FICO), CLTV, CUPB, origination rates, difference between origination and current interest rates, loan purpose, origination quarters, and loan age. Loan age refers to the elapsed time since loan origination up to September 2023 or until the time the loans were tracked, or until the payoff date (in the case that loans were paid off), whichever came first. Notably, loan age is not the number of months until default or until the time loans missed payments for 6 months. Borrower and property features included whether a borrower was a first-time home buyer, and whether properties were single-family homes or owner-occupied.

FICO is a strong indicator of borrowers' past credit behavior in making scheduled payments (9, 10), thus FICO was chosen as a predictive variable in identifying potential mortgage defaults. High CLTV is associated with high default risks since a high CLTV limits borrower equity and reduces incentive to make payments if property values decline (11). As demonstrated during housing market downturns, borrowers who had high CLTV strategically defaulted due to declining house prices and home equity. Higher mortgage interest rates correlate with higher default risks because high interest rates cause payment burdens and instability (12). Higher unpaid balances (CUPB) also imply higher loan amounts, resulting in increases in default probabilities and financial risks, especially during economic recession (13).

2.4 Correlation analysis

Correlation analysis is a core step to examine the relationship between borrower or loan characteristics and default behavior. This step is crucial in selecting appropriate features for developing accurate mortgage loan default prediction models.

A correlation heatmap was produced to help visualize the pairwise correlations among variables in the dataset (see Figure 1). A correlation value of 1 or -1 is a perfectly positive or negative correlation, whereas a correlation value of 0 implies no correlation. The correlation analysis and heatmap were constructed in Python using several Python libraries for data analysis and visualization. Specifically, the correlation matrix was generated using *df.corr()*, and visualized with *seaborn.heatmap*, while *matplotlib.pyplot* was used for plotting.

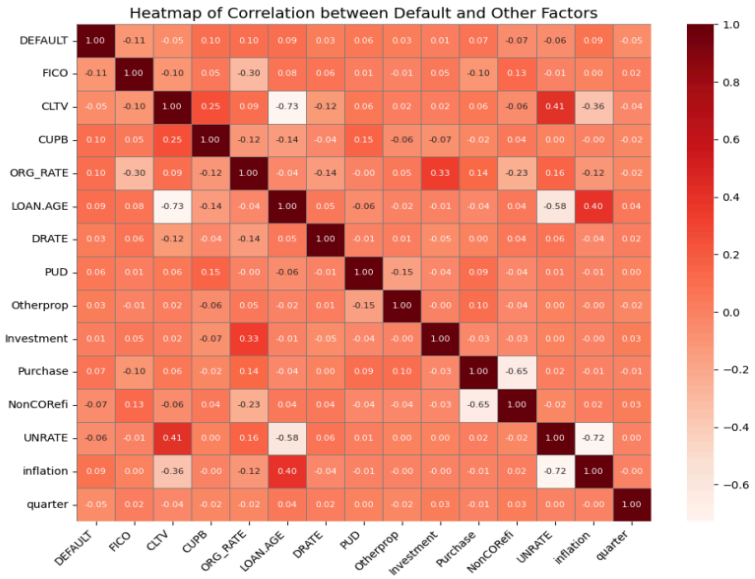

Figure 1. Correlation heat map between default status and other characteristics.

Table 1. VIF values for all features.

Variable	FICO	CLTV	CUPB	ORG_RATE	LOAN.AGE	DRATE	Otherprop	PUD	Investment	Purchase	NonCORefi	UNRATE	Inflation
VIF	1.16	2.36	1.13	1.42	2.79	1.06	1.04	1.06	1.17	1.79	1.82	2.79	2.15

The correlation heat map provides an effective way to visualize the directions and strength of relationships between default behavior and other features, as well as across features in the dataset. The directions of relationships between default behavior and other features were as expected. For example, default behavior was negatively correlated with higher FICO and positively correlated with CLTV and origination interest rates. This indicated that these loan, borrower characteristics, and macroeconomic factors were meaningful in classifying default behavior. It was also anticipated that the correlation values would be low because a Pearson correlation measures a linear relationship, and it cannot capture well the non-linear correlation between a binary default status and other features. ML models are better equipped mathematically to capture those non-linear relationships.

While some features showed moderately strong correlations (i.e., loan age and CLTV), this did not necessarily imply multicollinearity. Table 1 presents the variance inflation factor (VIF) values for each feature. All VIF values were < 5, indicating that there were no multicollinearity challenges, thereby ensuring optimal model performance and model evaluation.

3 Methods

ML is a scientific discipline focused on developing mathematical algorithms and statistical models that improve performance. The science of ML is closely connected to computational statistics, which facilitates more accurate predictions through a data-driven approach. ML allows for the identification of data patterns, insights, and valuable relationships in large datasets (14-16). Rather than relying on explicit programming, ML algorithms construct mathematical models from training data to make predictions on the testing data. Integrating ML algorithms into credit risk assessment plays an important role in predicting mortgage loan default accurately because it helps make a significant business impact in the real-world financial sector for large banks, mortgage lenders, financial regulators, investors, and policymakers.

To predict the probability of mortgage loan defaults, this study included three ML boosting methods as building blocks of tree-based classification techniques (17). Boosting algorithms rely on dependent ensembles, iteratively adding trained base models to reduce the misclassification of the current ensemble, which helps provide significantly better predictions than a single classifier in credit risk assessment (18). This section provides the detailed method of each boosting algorithm among Gradient Boosting, LightGBM, and XGBoost in comparison with Logistic Regression.

3.1 Gradient Boosting (GB) algorithm

Gradient Boosting is an algorithm that trains models to iteratively learn from previous errors and develops a more accurate model through ensemble techniques (19). The objective of the GB process is to minimize the residuals iteratively via a loss function and add new predictions until the final model has the highest accuracy. GB starts with an initial value (often the mean of the target variable) and generates a prediction for the first iteration. From there, the Gradient Boosting Model (GBM) makes predictions by gradually improving its estimate using residuals from the previous iterations. GB scales the estimate from the previous prediction using a learning rate and combines the predictions. Over several iterations, it can learn from its previous mistakes and thus obtain better predictions. GB uses a decision tree as the base leaner; however, it ensembles “weak learners” to learn the misclassification from each tree and add newly trained trees to reduce errors and decrease the overall loss. GBM shows the potential in classifying mortgage loan defaults due to its advantage of capturing non-linear and complex data patterns.

3.2 Light Gradient Boosting Machine (LightGBM) algorithm

LightGBM is an efficient and scalable boosting algorithm that uses histogram-based binning and a leaf-wise system to process large datasets with multiple features (20). LightGBM grows trees leaf-wise instead of level-wise like normal boosting algorithms. With the leaf-wise mechanism, Light GBM selects the leaf to be split with the maximum loss reduction. Therefore, this strategy allows trees to grow deeper, leading to the potential for increased accuracy.

This is especially true in binary classification problems. LightGBM also offers histogram-based feature binning. This reduces memory requirements and significantly increases the speed of computation.

The Gradient Boosting algorithm is divided into the following steps (19). Step 1 minimizes the loss function, given by
$G(x) = argmin_\theta \sum_{i=1}^{n} L(y_i, \theta)$ (eq 1), where y_i is the true value, and the loss function $L(y_i, \theta)$. Step 2 computes the residuals or the negative gradient of the loss function.

Step 3 fits a base learner and trains a decision tree on the residuals. Step 4 updates the model using the learning rate, given by
$G(x)_n = G(x)_{n-1} + \theta_n T(x_i; \alpha_n)$ (eq 2). Finally, Step 5 iteratively minimizes the loss function by adding newly trained trees to correct the residuals from previous trees. The recursive model is run until the convergence conditions are met.

The Light GBM's objective function is given by (20), $L^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t)$ (eq 3). where l is the loss function in which y_i is the actual default status and $\hat{y}_i$ is the predicted default status; $\Omega(f_t)$ is the additional regularization term to eliminate overfitting and model complexity. The Light GBM algorithm includes the following steps (20). Step 1 minimizes the regularized loss using the second-order Taylor series as the equation, $L^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2\right] + \Omega(f_t)$ (eq4), where g_i and h_i are first and second derivatives or Gradients and Hessians matrix, respectively. Step 2 computes the first and second derivatives, using both the Gradients and Hessians matrices. Step 3 uses histogram-based binning to increase computation speed. Features are binned into buckets to reduce computational time. Step 4 uses Leaf-Wise Tree Growth. LightGBM grows trees by splitting the leaf with the maximum loss reduction, which results in more accurate prediction. Finally, in Step 5 a tree is added to the ensemble and the model is updated iteratively.

LightGBM was developed based on using two key features: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). Effectively, GOSS enables LightGBM to focus more on larger gradients (often misclassified samples), leading to improved performance in binary classifications. The leaf-wise ability is prone to overfitting (particularly for smaller datasets) but can be mitigated through hyper-parameter tuning: max depth and minimum data, among other parameters. LightGBM is popular in the financial industry because it can achieve efficiency and prediction accuracy while reducing computational time significantly.

3.3 Extreme Gradient Boosting (XGBoost) algorithm

Extreme Gradient Boosting (XGBoost) is a supervised machine learning algorithm that ensembles classification trees with a scalable and regularized variant of Gradient Boosting Machines (21). It provides parallel tree boosting that offers high performance, time efficiency, and scalability. XGBoost expands upon the standard framework by considering a second-order Taylor approximation of the loss function and both first and second derivatives (i.e., Gradient and Hessian) for function optimization.

XGBoost provides stable improvements through learning at each iteration and handling non-concavity efficiently to achieve refined performance. The main advantage of XGBoost is its capability to regulate model complexity and prevent overfitting using regularization techniques. XGBoost incorporates additional improvements, such as a) column block store to allow parallel computation, b) sparsity-aware algorithms to effectively handle missing values, and c) loss-based tree pruning to increase model performance. Overall, these optimizations help XGBoost outperform many established machine learning algorithms, offering high accuracy and fast performance.

XGBoost offers the best performance in many cases due to its advanced boosting techniques and regularization, though it requires careful parameter tuning to avoid overfitting. XGBoost also combines weak learners to create a strong predictive model. It has acquired a reputation as being one of the fastest gradient boosting algorithms. XGBoost mitigates the inefficiency due to evaluating losses of all possible splits by examining the distribution of features across all data points in a leaf, thereby narrowing the search space for potential splits. This inefficiency is further alleviated when the number of inputs increases in a large dataset.

XGBoost's speed is its most advantageous feature. This rapid performance allows for the efficient exploration of numerous hyperparameter settings, which is essential given the large number of hyperparameters that require tuning. Most of these hyperparameters are aimed at preventing overfitting, as combining thousands of base models can easily lead to overfitting despite their simplicity. XGBoost minimizes a regularized loss, making it robust to overfitting. Based on its desired advantages, XGBoost has been used extensively in credit risk modeling for commercial banks to provide accurate loan default predictions.

The XGBoost's objective function is given by (21), $XGB = \sum_{i=1}^{n} L\left(y_i, f(x_i)\right) + \sum_{k=1}^{t} \Omega(f_k)$ (eq 5), where $\Omega(f_k)$ is the regularization term at the kth iteration, which is expressed as the following, $\Omega(f_k) = \beta T + \frac{1}{2}\rho \sum_{j=1}^{J} w_j^2$ (eq 6), where β is the complexity of leaves, T is the number of leaves, ρ denotes the penalty parameter, and w_j is the weight of each leaf node j.

The objective function for XGBoost is similar to LightGBM but the two models are different in terms of efficiency techniques and the tree building method. XGBoost grows trees level-wise and uses continuous features instead of binning features.

The XGBoost algorithm includes the following steps (21). Step 1 minimizes regularized Loss using second-order Taylor series as,
$L^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t)$
(eq7), where g_i and h_i are the first and second derivatives, represented by the Gradients and Hessians matrices respectively. Step 2 computes the first and second derivatives, using both the Gradients and Hessians matrices. Step 3 fits a tree to Gradient Statistics, and maps inputs to values that minimize the loss function. Step 4 computes the optimal leaf weights and finally Step 5 adds a tree to the ensemble and updates the model iteratively. Overall, with the introduction of regularization terms, XGBoost is more effective at preventing overfitting while achieving better prediction performance.

3.4 Logistic Regression (LR)

Traditional credit scoring models utilize logit models, as binary classification models (3, 22). The probability of default status is modeled using a sigmoid function (see equation 8). The log-odds of default are expressed as a linear combination of borrower characteristics (X) and their corresponding coefficients (β) (see equation 9). $P(Y_i = 1) = \frac{e^{X\beta}}{1+e^{X\beta}}$ (eq 8) $log\left(\frac{P(Y_i=1)}{1-P(Y_i=1)}\right) = X\beta$ (eq 9)., where $X\beta = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_3 X_3 + e$.

Even though the logit model results can be interpreted intuitively, linearity and independence assumptions are strong model assumptions. Boosting algorithms offer a more flexible and powerful framework that can capture complex, nonlinear relationships in high-dimensional datasets of rich borrower, loan, and property characteristics (4, 18). This makes ML boosting algorithms especially more innovative, accurate, and informative in credit decisions in practice.

4 Building mortgage loan default models

4.1 Data modeling and cross-validation

The data modeling process with the ML boosting algorithms in this paper was structured as shown in Figure 2.

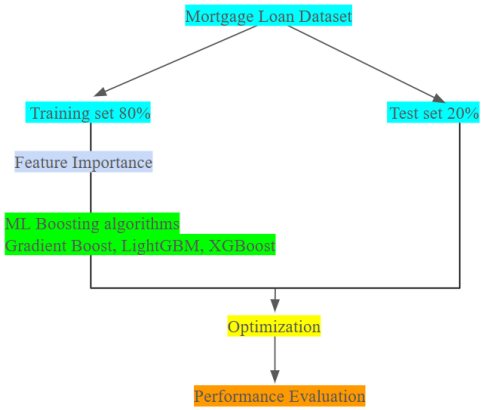

Figure 2. Mortgage loan default data modeling diagram.

The balanced data was stratified split into 80% for training and 20% for testing using the Python *train_test_split* function. The data modeling process also employed Python libraries, including Scikit-Learn, to perform data preprocessing and model evaluation. The training data was used to train the model, and the final trained model was applied to the testing data to obtain predictions of mortgage loan defaults.

4.1.1 *K-fold validations*

K-fold cross-validation is a resampling technique for evaluating the generalizability and robustness of predictive models. The key benefit of cross-validation methods is to evaluate model performance on new data. One of its benefits is to reduce the overfitting risk that often arises in the presence of high-dimensional or noisy datasets. Using this approach, the training data were randomly partitioned into K equal subsets, or "5 folds". In each iteration, (K -1) folds (i.e., four folds in 5-fold cross-validation) were used to train the model, while the remaining fold was held out for validation.

This process was repeated K times. The results from all iterations were ultimately averaged to produce an overall estimate of the model. A 5-fold cross-validation procedure was employed to evaluate all four classification models in this study. The 5-fold approach provides a balance between computational efficiency and reliable performance estimation.

Four classification algorithms, including Logistic Regression and the three boosting algorithms: Gradient Boosting, XGBoost, and LightGBM, were used to train the predictive models. The entire calculation process was performed in R and Python. R libraries with *xgboost*, *lightgbm*, and *gbm* packages were used for ML Boosting algorithm implementation. Supporting libraries, such as caret for model evaluation and *ggplot2* for visualization, were also applied throughout the analysis. The Python's Scikit-Learn, Gradient Boosting, XGBoost, and LightGBM libraries, respectively, were also employed in this analysis. Python's Scikit-Learn is the primary ML library. In addition, the Python package Matplotlib was used to perform data distribution analysis for multiple features. The paper also employed other libraries, including the NumPy library for working with matrices and math operations, the SciPy library for scientific and technical computing, the Matplotlib library for data visualization, and the Pandas library for data handling, manipulation, and analysis.

4.2 Feature importance

Feature Engineering (FE) is a valuable tool in ML to identify those features that contribute the most to model predictions. FE enhances model interpretability, supports feature selection by giving the influential size of each predictor so that key drivers of outcomes can be recognized, trained, and used for improving model accuracy.

After feature extraction and selection, the boosting models were fitted based on the training set. The test set was adopted to evaluate model performances across boosting models and LR. Hyper-parameter tuning was applied to examine whether model performance improved and to obtain optimal performance. The last stage was to compare the performance of these models before and after optimization; followed by the determination of the optimal model and its deployment.

Table 2. Hyper-parameter optimization of the boosting algorithms.

Hyper-parameters	**Description**	**Ranges of Parameters**	**Optimal Value**
Learning Rate	Shrinking coefficient of each tree	0.001, 0.01, 0.05, 0.1, 0.5	0.1
Max Depth	Maximum depth from the root to the leaf of a tree.	3, 5, 9, 11, 15, 19	9
Number of leaves	Number of leaves for each tree	15, 20, 30, 40	30
Max Features	Proportion of randomly selected features each iteration	0.5, 1	1
Sample split	The subsample rate of features for every split each tree	0.1, 1	1
Number of estimators	The highest number of base learners	100, 500, and 1000	1000

4.3 Hyper-parameter optimization

The performance of boosting algorithms depends on tuning the hyper-parameters to achieve superior performance (23). The hyper-parameters that contributed most significantly to the robustness and accuracy of the model were the number of estimators, learning rate, max features, max depth, min samples split, min samples leaf for Light GBM; in addition to *min_child_weight*, and *colsample_by_tree* for XGBoost.

Table 2 shows the hyper-parameters that were tuned during the optimization process of the boosting algorithms. The learning rate controls the contribution of each individual tree to the overall ensemble by shrinking the weights assigned at each boosting step. A lower learning rate typically enhances model robustness, reduces the risk of overfitting; however, excessively small values can lead to under-fitting and long training time. Conversely, a high learning rate may speed up convergence but can increase the risk of overfitting. A higher number of estimators is associated with the better model performance. However, the higher number of estimators can increase computation cost and cause model complexity. Deeper trees can capture complex nonlinear patterns, but they may also over-fit the training data, whereas shallower trees may underfit. Higher values of the maximum number of features increases correlation between trees in the ensemble, may encounter over-fitting, while lower values introduce more randomness, helping to reduce inter-tree correlation and enhance generalization. The sample split serves as a regularization parameter, preventing the model from learning spurious patterns when set at higher values.

Overall, these hyper-parameters define the trade-offs between learning speed, accuracy, variance, and computational efficiency. If the value of the parameter is set too low, there might be an underfit prediction. In contrast, if the value of the parameter is set too high, the computation costs increase. The lower values may alleviate the over-fitting challenge while the higher values may cause under-fitting. Thus, optimal tuning is essential to ensure that the boosting algorithm generalizes well across new and unseen data. After optimization, the last column in Table 2 shows the final results of selected parameters that balanced between prediction accuracy, overfitting, and computational time.

4.4 Model performance evaluation

In this research, credit scoring machine learning models were evaluated based on four machine learning algorithms: Logistic Regression, Gradient Boosting, LightGBM, and XGBoost, using the results from both training and testing data. Model performance evaluation across ML boosting models and LR was performed for both the training and the testing datasets. In order to assess whether the model was overfitting, it is important to compare testing accuracy with training accuracy across models to see if extremely poor performance in testing data occurs. Table 3 presents model performance metrics used to evaluate the accuracy of ML boosting prediction algorithms compared to the Logistic Regression model. Each of the metrics is specified with its associated formula and description in the table.

Table 3. Model performance metrics.

Metric	Description	Mathematical Formula
ACC	Accuracy	$ACC = (TP + TN)/(TP + FP + TN + FN)$
SEN	Sensitivity	$SEN = TP/(TP + FN)$
SPE	Specificity	$SPE = TN/(FP + TN)$
TPR	True positive rate	$TPR = TP/(TP + FN)$
FPR	False positive rate	$FPR = FP/(FP + TN)$
F1	F1 Score	$F1 = 2\,(Precision * Recall)/(Precision + Recall)$
AUC	Area under the curve	$AUC = (1 + TPR - FPR)/2$

4.4.1 *Accuracy*

Accuracy refers to the overall correctly predicted probabilities of both positive and negative outcomes (defaults and non-defaults). In credit risk assessment, a higher accuracy implies a higher share of correct classification of loan defaults and loan repayments relative to the total number of loans in the sample.

4.4.2 *Confusion Matrix and ROC*

The confusion matrix serves as a basic statistical tool for classifying a binary outcome, which can be used to determine the accuracy of ML models that predict loan default statuses. A confusion matrix is used to compare the prediction to the actual default and evaluate if "good" or "bad" loans are predicted correctly. The four cells in the confusion matrix represent True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). Within this table, two important performance measures are contained with relation to the confusion matrix, the true positive rate (TPR) and false positive rate (FPR). The TPR and FPR are two key metrics in forming the Receiver Operating Characteristic (ROC) curve (24).

TPR represents the ratio of actual positive observations, such as loans that defaulted, that were correctly identified by the model. This is also known as recall or sensitivity and describes the model's ability to correctly detect loan defaults out of all actual defaults. Meanwhile, FPR is the proportion of actual negative outcomes, such as loans that did not default and loans that were incorrectly identified as defaults. This measure indicates the extent to which the model creates false alarms by identifying repaid loans as defaulted loans.

Sensitivity measures the model's ability to correctly identify actual defaults (TP), while specificity reflects its ability to correctly identify repaid loans (TN). The Recall metric is calculated in the same way as Sensitivity or TPR. Precision is calculated as TP/(FP + TP). Using only precision or recall does not completely capture the prediction accuracy of various models in terms of ranking. Therefore, as a harmonic mean between Recall and Precision, F1 Score is used as another metric to rank the model accuracy across ML models.

The Area under the ROC Curve (AUC) indicates how well a model can distinguish between positive and negative outcomes (e.g., "bad" and "good" loans). A larger AUC value indicates that the model is better at predicting mortgage loan defaults across a range of classification thresholds. On the other hand, AUC values that approach 0.5 indicate a poor level of discrimination or that the model's predictions are no better than random guesses. Finally, AUC values that approach 1.0 have better predictive power. While ROC tends to show the trade-off between the TPR and FPR at multiple thresholds, AUC serves as a key overall measurement statistic of the model's classification accuracy.

5 Results and discussion

In this research, credit scoring machine learning models were evaluated based on four machine learning algorithms: Logistic Regression, Gradient Boosting, LightGBM, and XGBoost, using the results from both training and testing data. Performance evaluation metrics include Accuracy, AUC, F1 Score, Sensitivity and Specificity, TPR and FPR, or ROC. The prediction results after optimization showed that XGBoost and Light GBM outperformed the results from Logistic Regression, and Gradient Boosting. XGBoost and Light GBM were the best models after optimization, achieving 98% accuracy on both the training and testing data.

5.1 Results from Feature Importance

Feature Importance analysis found consistent patterns across the three boosting models (Figure 3). The top predictors of mortgage loan default from the GBM were loan age, CLTV, CUPB, origination interest rates, FICO, and unemployment rate. In the LightGBM model, CLTV ranked highest, followed by loan age, unemployment rate, CUPB, origination rates, and FICO, in addition to inflation rates. Similarly, the XGBoost model also ranked CLTV as the top predictor, followed by loan age, CUPB, unemployment rates, FICO, inflation, and origination interest rates as the most influential variables. These results suggested that creditworthiness indicators remained the most significant predictors of loan default across the different boosting techniques.

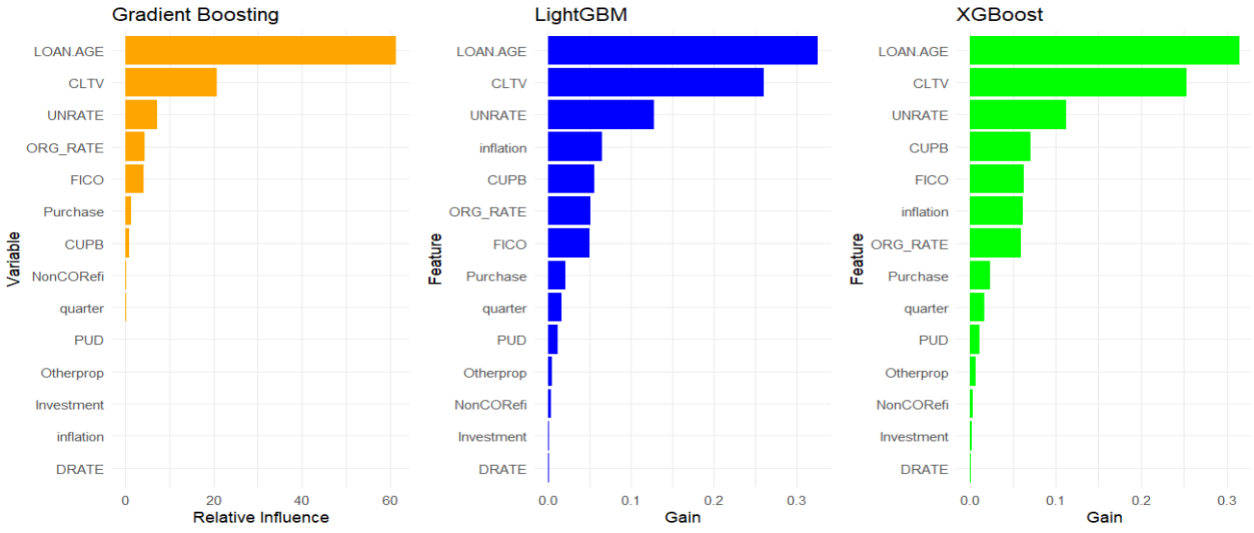

Figure 3. Feature Importance in mortgage loan default modeling.

5.2 Results from confusion matrix

A confusion matrix is a standard statistical tool to validate the correctly predicted loan defaults against actual loan defaults for the classification model. The matrix contains four quadrants: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). A higher total of TP and TN is associated with a higher model accuracy.

Figure 4 shows the comparison of the confusion matrix for four ML models, indicating that Light GBM and XGBoost were the best models with accurate predictions of loan defaults (i.e., more than 6400 loans). Both TP and TN for Light GBM and XGBoost were significantly greater than those for Logistic Regression and GBM, based on the prediction results of the testing data after optimization.

Based on the confusion matrix, classification performance could be further assessed using F1 Score, sensitivity (TPR) and specificity (TNR), TPR and FPR on the testing data. Among the four models: Logistic Regression, GBM, LightGBM, and XGBoost, XGBoost achieved the highest F1 Score (0.982) in the testing data, indicating a better balance between precision and recall (see Table 4). LightGBM achieved an F1 Score of 0.981, indicating performance close to that of XGBoost. Followed by XGBoost and LightGBM, GBM and LR presented with significantly lower F1 Scores of 0.757 and 0.685, respectively.

XGBoost achieved the highest specificity (0.983), whereas LightGBM obtained the highest sensitivity (0.981) in the testing data. This suggested that XGBoost was slightly better suited for minimizing FP (112 loans), whereas LightGBM performed slightly better in minimizing FN (125 loans). Both XGBoost and LightGBM achieved specificity and sensitivity ~ 98%, indicating they were most effective in measuring the probability of loan defaults and non-defaults correctly among the four models. Followed XGBoost and LightGBM, GBM and LR presented with lower specificities (0.742 and 0.656 respectively), indicating weaker performance in identifying non-default loans. GBM and LR also presented with lower sensitivities (0.805 and 0.7 respectively), demonstrated by a significant amount of FN in the confusion matrix (1278 and 1967 loans that were misclassified as non-defaults). Due to their limitations in capturing nonlinearity and special data patterns in a large data, GBM and LR had lower F1 scores, sensitivity and specificity than XGBoost and LightGBM.

The results also indicated that XGBoost and LightGBM yielded the highest accuracy or the lowest error with TPR of 98% and FPR of ~ 2%, indicating that XGBoost and LightGBM were the top performers in classifying loan defaults. Following XGBoost and LightGBM was GBM with a TPR of 0.805 and FPR of 0.258. LR had the lowest TPR of 0.7 and the highest FPR of 0.345, indicating the weakest performance among the four models.

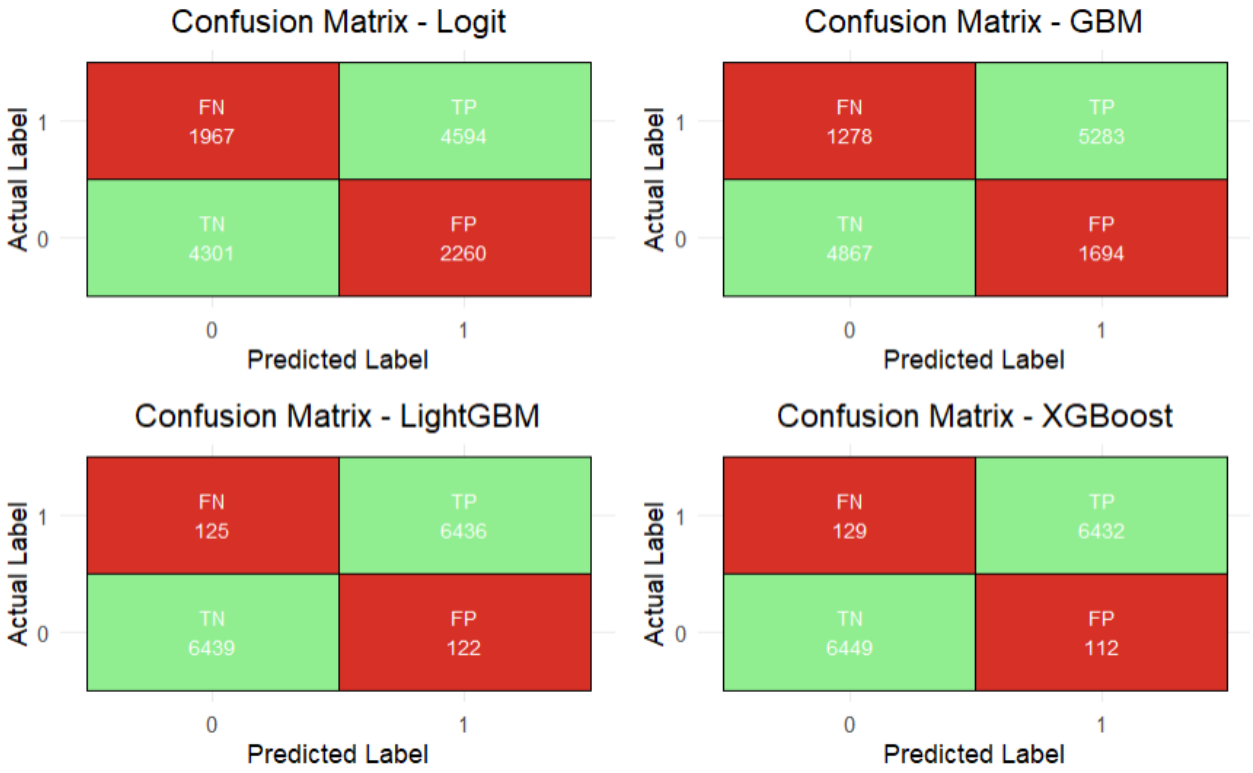

Figure 4. Comparison of confusion matrix for four ML models: Logistic Regression, Gradient Boosting, Light GBM and XGBoost – testing data.

5.3 Accuracy

The results from Table 4 indicate that XGBoost achieved the highest accuracy on both training and testing data (98.2%). In addition, both XGBoost and LightGBM performed better on the training data and testing data after hyper-parameter tuning optimization. Specifically, XGBoost achieved 98.2% accuracy and LightGBM achieved 98.1% accuracy on the testing data. Meanwhile, GBM and LR had relatively lower accuracy rates (76% and 68%, respectively) compared to XGBoost and LightGBM in the testing data.

Table 4. Model performance results for LR, GB, LightGBM and XGBoost ML algorithms.

Model	**Logistic Regression**		**Gradient Boosting**		**Light GBM**		**XGBoost**	
Metrics	***Training***	***Testing***	***Training***	***Testing***	***Training***	***Testing***	***Training***	***Testing***
Accuracy	0.665	0.678	0.756	0.774	0.981	0.981	0.982	0.982
AUC	0.665	0.678	0.756	0.774	0.981	0.981	0.982	0.982
F1 Score	0.669	0.685	0.758	0.781	0.981	0.981	0.982	0.982
Sensitivity	0.676	0.700	0.766	0.805	0.979	0.981	0.980	0.980
Specificity	0.655	0.656	0.745	0.742	0.982	0.981	0.983	0.983
TPR	0.676	0.700	0.766	0.805	0.979	0.981	0.980	0.980
FPR	0.345	0.345	0.255	0.258	0.018	0.019	0.017	0.017

XGBoost performed slightly better than LightGBM in terms of accuracy, AUC, F1 score, specificity and FPR because XGBoost could accurately predict loan defaults and had a lower misclassification rate of loan defaults than LightGBM. Even though both XGBoost and LightGBM were the best performers in predicting mortgage loan defaults after optimization, with better performance on all evaluation metrics, XGBoost had the best performance with the best overall model performance among all ML boosting algorithms. The overall results also demonstrated that the ML boosting algorithms were more accurate than the traditional regression-based approach.

Table 5 shows the detailed 5-fold cross validation results for LR, GB, LightGBM and XGBoost algorithms, demonstrating that XGBoost was the most accurate across 5 folds and the best overall performing model among all ML algorithms.

Table 5. The 5-fold cross validation results for LR, GB, LightGBM and XGBoost algorithms.

Model	**Logistic Regression**		**Gradient Boosting**		**Light GBM**		**XGBoost**	
Metrics	***Mean***	***SD***	***Mean***	***SD***	***Mean***	***SD***	***Mean***	***SD***
Overall Accuracy	0.665	0.0041	0.756	0.0046	0.981	0.0015	0.982	0.0012
Fold 1 Accuracy	0.666	0.0023	0.759	0.0034	0.981	0.0017	0.982	0.0014
Fold 2 Accuracy	0.665	0.0025	0.757	0.0032	0.981	0.0019	0.983	0.0013
Fold 3 Accuracy	0.663	0.0031	0.755	0.0068	0.983	0.0005	0.983	0.0009
Fold 4 Accuracy	0.667	0.0063	0.752	0.0047	0.980	0.0014	0.981	0.0007
Fold 5 Accuracy	0.665	0.0053	0.755	0.0026	0.981	0.0015	0.982	0.0009

While XGBoost consistently achieved the highest overall model accuracy, LightGBM demonstrated the most substantial gains in accuracy after hyper-parameter optimization. Although feature importance highlights which predictors contributed most to model accuracy (Figure 3), the empirical findings demonstrated that hyper-parameter tuning played an even more critical role in enhancing overall model performance, with particularly substantial gains in predictive performance observed for LightGBM when comparing pre- and post-optimization results (Table A.1 for pre-optimization results in the Appendix and Table 4 for post-optimization results for comparison). The difference in performance gains between LightGBM and XGBoost after hyper-parameter tuning or optimization, despite similar feature importance rankings, can be explained by the underlying algorithmic differences between the two models.

While both are gradient boosting frameworks, LightGBM uses a leaf-wise (best-first) tree growth strategy with depth constraints, whereas XGBoost typically grows trees level-wise, as discussed in Section 3.2 and Section 3.3. on LightGBM and XGBoost model methodology.

The leaf-wise strategy tends to produce more complex interactions between features and can yield higher accuracy when the model is carefully tuned. However, it is also more sensitive to parameter settings such as number of leaves, learning rate, depth of the tree, which explains why optimization produces a larger relative performance improvement for LightGBM than for XGBoost.

In contrast, XGBoost's level-wise approach is more conservative and less sensitive to parameter tuning, meaning the gains from hyper-parameter optimization are typically smaller. Importantly, feature importance measures only the relative contribution of features to splits, not the efficiency of the boosting process itself. Thus, two models can have similar feature importance profiles (CLTV, CUPB, etc.) but differ in how well their boosting algorithms (i.e., parameter settings) leverage those features under optimized settings.

5.4 AUC and ROC performance

XGBoost achieved the highest prediction accuracy in predicting mortgage loan defaults with an AUC of 0.982 in both training and testing data. LightGBM also achieved similar AUC as XGBoost, with slightly lower AUC in both training and testing data (0.981). Following XGBoost and LightGBM was GBM with AUC of 0.774 in the testing data. Meanwhile, the regression-based approach with the logit model only has an AUC of 0.678 in the testing data, indicating the poorest performance among the four models. XGBoost and Light GBM outperform GBM and LR significantly in terms of AUC on both training and testing data.

Figure 5 compares the ROC curves across ML boosting algorithms and LR in this study. XGBoost and Light GBM achieved the best prediction accuracy in mortgage loan defaults prediction in both training and testing data. The Logit model demonstrated potential in classifying loan defaults, however, with the lowest predictive accuracy. GBM performed slightly better than the Logit model but was hampered by much lower accuracy than LightGBM and XGBoost. Overall, the ML boosting algorithms demonstrated significant improvement in predicting loan defaults. XGBoost and Light GBM models were the most accurate.

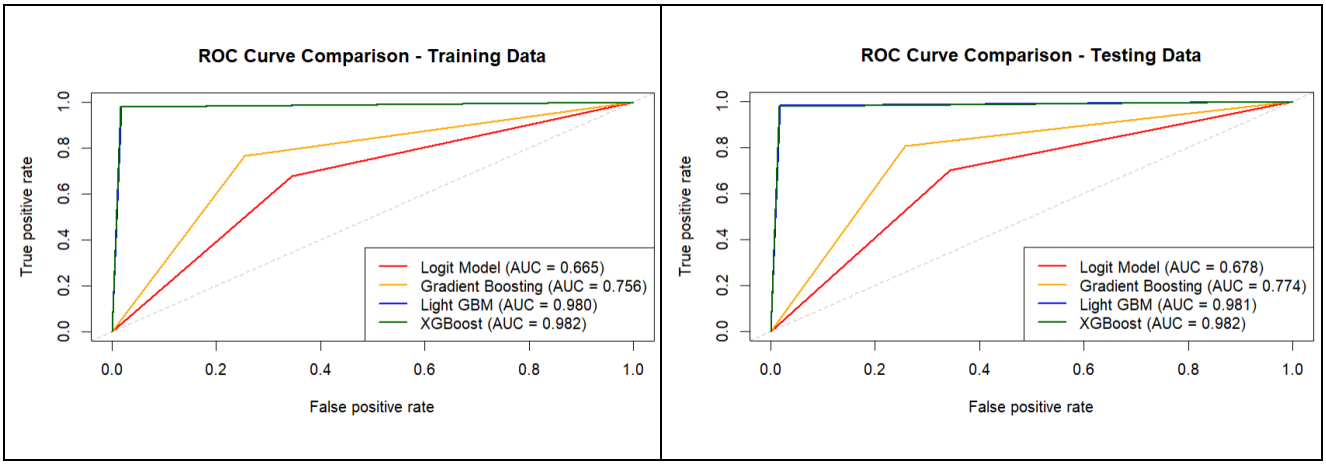

Figure 5. ROC and AUC comparison for four ML models: Logistic Regression, Gradient Boosting. Light GBM and XGBoost – training and testing Data. Figure 5a and 5b shows the ROC curve comparison on the training data and the testing data, respectively.

Overall model performance assessment indicated that XGBoost and LightGBM outperformed Logistic Regression and GBM in predicting U.S. mortgage loan defaults. Boosting algorithms possessed higher accuracy, higher F1 scores, balanced recall and precision metrics, higher sensitivity, specificity and higher AUC scores than traditional regression-based models. Logistic Regression, while interpretable and meaningful, yielded the poorest performance in identifying "bad" loans, or mortgage loan defaults.

ML Boosting models can predict mortgage loan defaults more accurately than the traditional Logistic Regression approach. Boosting models successfully minimized the inaccuracy because they could account for various relationships, nonlinear and concave patterns in the dataset with multiple features that were often overlooked by simpler models such as the Logistic Regression model. These results demonstrate that ML boosting algorithms, including XGBoost and LightGBM, were more effective in capturing complex patterns in borrower behavior, making them an ideal choice for financial institutions managing credit risks.

5.5 Perspectives

Future studies should analyze other ML algorithms including deep learning, neural network and other tree-based algorithms. Based on the findings of this paper, ML Boosting models including XGBoost and LightGBM can be used as effective baseline models for comparison.

Future research could extend the current study by examining loan performance under historical or simulated economic stress scenarios. The dataset used in this study comprised of loans originated in 2020, with performance observed through September 2023, and therefore did not include loans from earlier periods, such as the 2007–2010 subprime mortgage crisis. While historical risk mispricing during that period is informative for understanding past systemic vulnerabilities, it is not directly relevant to the more recent underwriting standards and macroeconomic conditions reflected in the current data. To enhance the robustness and generalizability of the proposed models, future work could add stress tests using simulated environments or historical scenarios characterized by extreme conditions, such as high inflation rates exceeding 10% or characteristics of subprime mortgage loans, to support more resilient risk assessment frameworks for banks, financial institutions, and regulators.

6 Conclusion

The study showed that ML boosting algorithms, including Gradient Boosting, XGBoost, and Light GBM, outperformed Logistic Regression in predicting mortgage loan defaults. The paper found that XGBoost and Light GBM were the top-performers with 98% accuracy on the testing data. The results were based on Freddie Mac data, a rich set of loan, borrower and property characteristics, combined with boosting algorithms with regularization techniques and hyper-parameter tuning within a data-driven framework. The finding supported the key hypothesis that ML boosting algorithms, especially XGBoost and Light GBM, were more accurate compared to the traditional Logistic Regression model in predicting mortgage loan defaults.

These ML models, XGBoost and Light GBM, were well-positioned to capture nonlinear patterns in borrower credit profiles simultaneously with the macroeconomic landscape to improve model prediction accuracy. Although the ML boosting models provided improved accuracy, traditional models such as Logistic Regression are likely to remain beneficial for interpreting the impacts of specific risk factors and economic indicators.

The study can be used as guidance for ML algorithm selection and designing a system that can be used to predict loan defaults in credit risk assessment for large banks and the financial industry. This would help identify "bad" loans in a timely manner and minimize potential loss. Future applications of this research would consist of applying the same techniques in the prediction of binary outcomes in any field. AI and ML tools will help professionals make better predictions, reduce errors, and achieve more effective and accurate decisions.

The finding contributes a significant solution to the FinTech industry, including banks, mortgage lenders, originators, and regulators. Beyond the context of mortgage lending, the paper also contributes to a broader science community with highly accurate classification models.

Acknowledgements

I would like to thank my AP Statistics, Computer Science teachers, online Harvard and Stanford course instructors in Data Science, Machine Learning and AI with R, Python for helping me discover my strong interests in data science, finance, statistics, AI/ML models, and innovative technology. My sincere thanks to Professor Henisz Witold and Professor Anne Jamison from the University of Pennsylvania for inspiring my research interests in finance and the FinTech industry.

7 References

1. Thomas, L. C., Crook, J. N., and Edelman, D. B. (2017). *Credit scoring and its applications, second edition*. Society For Industrial And Applied Mathematics. https://doi.org/10.1137/1.9781611974560

2. Mian, A., and Sufi, A. (2010). The Great Recession: Lessons from Microeconomic Data. *American Economic Review*, 100(2), 51–56. https://doi.org/10.1257/aer.100.2.51

3. Hand, D. J., and Henley, W. E. (1997). Statistical Classification Methods in Consumer Credit Scoring: a Review. *Journal of the Royal Statistical Society: Series a (Statistics in Society)*, 160(3), 523–541. https://doi.org/10.1111/j.1467-985x.1997.00078.x

4. Khandani, A. E., Kim, A. J., and Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11), 2767–2787. https://doi.org/10.1016/j.jbankfin.2010.06.001

5. Lessmann, S., Baesens, B., Seow, H.-V., and Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 247(1), 124–136. https://doi.org/10.1016/j.ejor.2015.05.030

6. Freddie Mac. (2025). Single Family Loan-Level Dataset. Www.freddiemac.com. https://www.freddiemac.com/research/datasets/sf-loanlevel-dataset

7. Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16(16), 321–357. https://doi.org/10.1613/jair.953

8. Goel, A., and Rastogi, S. (2021). Understanding the impact of borrowers' behavioural and psychological traits on credit default: review and conceptual model. *Review of Behavioral Finance*, 15(2). https://doi.org/10.1108/rbf-03-2021-0051

9. Demyanyk, Y. S., Ralph, & Hemert, O. V. (2010). Determinants and Consequences of Mortgage Default. *Federal Reserve Bank of Cleveland*, *Working Paper No. 10-19*. https://doi.org/10.26509/frbc-wp-201019

10. Sengupta, R., and Bhardwaj, G. (2015). Credit Scoring and Loan Default. *International Review of Finance*, 15(2), 139–167. https://doi.org/10.1111/irfi.12048

11. Otero González, L., Durán Santomil, P., Lado Sestayo, R., and Vivel Búa, M. (2016). The impact of loan-to-value on the default rate of residential mortgage-backed securities. *The Journal of Credit Risk*, 12(3). https://doi.org/10.21314/jcr.2016.210

12. Ahmad, F., & Choudhry Tanveer Shehzad. (2024). The role of interest rate environment in mortgage pricing. *International Review of Economics & Finance*, 89, 225–245. https://doi.org/10.1016/j.iref.2023.07.102

13. Qi, M., and Yang, X. (2009). Loss given default of high loan-to-value residential mortgages. *Journal of Banking & Finance*, 33(5), 788–799. https://doi.org/10.1016/j.jbankfin.2008.09.010

14. Dixon, M. F., Halperin, I., & Bilokon, P. A. (2020). *Machine Learning in Finance: from Theory to Practice* (1st ed.). Springer. https://doi.org/10.1007/978-3-030-41068-1

15. Gogas, P., and Papadimitriou, T. (2021). Machine Learning in Economics and Finance. *Computational Economics*, 57(1), 1–4. https://doi.org/10.1007/s10614-021-10094-w

16. Lopez De Prado, M. (2018). *Advances in Financial Machine Learning*. Wiley. https://dl.acm.org/doi/10.5555/3217448

17. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R* (2nd ed.). Springer. https://doi.org/10.1007/978-1-0716-1418-1

18. Finlay, S. (2011). Multiple classifier architectures and their application to credit risk assessment. *European Journal of Operational Research*, 210(2), 368–378. https://doi.org/10.1016/j.ejor.2010.09.029

19. Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

20. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., et al. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30, 3146–3154.

21. Chen, T., and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, 785–794. https://dl.acm.org/doi/pdf/10.1145/2939672.2939785

22. Hand, D. J. (2001). Modelling consumer credit risk. *IMA Journal of Management Mathematics*, 12(2), 139–155. https://doi.org/10.1093/imaman/12.2.139

23. Hoof, J. van, and Vanschoren, J. (2021). Hyperboost: Hyperparameter Optimization by Gradient Boosting surrogate models. *ArXiv* (Cornell University), *abs/2101.02289*. https://doi.org/10.48550/arxiv.2101.02289

24. Nellore, S. B. (2015). Various performance measures in Binary classification-An Overview of ROC study. *IJISET-International Journal of Innovative Science, Engineering & Technology*, 2(9), 596–605.

Appendix

Table A1. Model performance results for LR, GB, LightGBM and XGBoost ML algorithm before hyper-parameter optimization.

Model	**Logistic Regression**		**Gradient Boosting**		**Light GBM**		**XGBoost**	
Metrics	***Training***	***Testing***	***Training***	***Testing***	***Training***	***Testing***	***Training***	***Testing***
Accuracy	0.665	0.678	0.756	0.774	0.843	0.852	0.966	0.965
AUC	0.665	0.678	0.756	0.774	0.843	0.852	0.966	0.965
F1 Score	0.669	0.685	0.758	0.781	0.850	0.861	0.966	0.965
Sensitivity	0.676	0.700	0.766	0.805	0.894	0.913	0.962	0.959
Specificity	0.655	0.656	0.745	0.742	0.791	0.791	0.969	0.972
TPR	0.676	0.700	0.766	0.805	0.894	0.913	0.962	0.959
FPR	0.345	0.345	0.255	0.258	0.209	0.209	0.031	0.029