Enhancing Credit Risk Prediction: A Comparative Study of Machine Learning Techniques with Advanced Feature Engineering

Binh Son Nguyen, Timothy Pasaribu

November 30, 2024

1 Introduction

The primary goal of this project is to develop a robust credit risk model that accurately predicts the likelihood of whether a borrower will pay the loan on time. Aside from using traditional models learned in the course, We will put an emphasis on exploring gradient boosting techniques to harness their superior performance in handling complex, non-linear relationships within the data. This exploration aims to understand the strengths of gradient boosting in credit risk scenarios and assess its effectiveness compared to traditional models.

2 Literature Review

Recent advancements in credit risk modeling using machine learning, particularly gradient boosting techniques, have shown promising results. Tian et al. (2020) [3] demonstrated the effectiveness of Gradient Boosting Decision Trees (GBDT) in predicting loan defaults, achieving high accuracy, F1 scores, and AUC values, making it well-suited for financial data analysis.

Similarly, Chang et al. (2018) [1] explored eXtreme Gradient Boosting (XGBoost) for handling imbalanced datasets in credit risk analysis. They found that XGBoost outperformed traditional classifiers like logistic regression and SVM, particularly in managing skewed class distributions common in financial datasets.

Practical implementations of credit risk models are also available on open-source platforms. Li (2023) [2] provided a framework for loan default prediction using Lending Club data, including preprocessing techniques such as median and mode imputation for missing values, and one-hot encoding for categorical features. These methods are valuable for building machine learning pipelines tailored to financial datasets.

Building on these works, our project compares the performance of ensembling methods like Random Forest and Gradient Boosting with traditional models like logistic regression. We introduce a novel feature engineering approach using anomaly scores to capture subtle risk factors, offering a new perspective on credit risk modeling.

3 Data

3.1 Data Description

We use LendingClub Loan Data (Kaggle - LendingClub) to train our models. The data consists of 2 datasets, the accepted loan dataset and the rejected loan dataset. This accepted dataset encompasses detailed information on loan applications, borrower demographics, loan terms, and repayment outcomes. It includes variables such as loan amount, interest rate, borrower credit score, employment length, and loan status (fully paid, current, charged off, etc.), providing a comprehensive basis for modelling credit risk. The accepted data is the one we use for model building. The rejected data contains fewer features and is used for feature engineering.

3.2 Data Preprocessing & Exploration

We conducted exploratory data analysis (EDA) to examine variable distributions, correlations, and patterns related to loan repayment. Loans with labels such as "Current" and "Grace Period" were excluded, as they were not relevant to our prediction goal.

To ensure data quality, we removed features with more than 40% missing values, which could reduce interpretability and predictive power. We also excluded "cheat features," like late fees and collection costs, which would not be available during the loan approval process. Variables like "URL," "emp_title," and "zip_code" were removed based on domain knowledge, as they were deemed irrelevant to credit risk prediction.

We addressed multicollinearity by removing one feature from any pair with a high correlation (e.g., Pearson > 0.8). Missing values were imputed using the median for numerical features and the mode for categorical features. Categorical variables were one-hot encoded for compatibility with machine learning models. These steps ensured that the dataset was clean, meaningful, and ready for modeling.

4 Problem Formulation

We will run a combination of different models to determine the best models among the models ran. As a baseline, we will start with logistic regression, followed by a standard weighted logistic regression variant to address the significant class imbalance found in our dataset. The next step involves employing ensemble methods, starting with random forests, which use bagging to capture complex feature interactions and improve predictive stability. We will explore gradient boosting techniques using XGBoost. We aim to compare the performance of models both with and without the inclusion of the newly engineered feature to evaluate its impact on predictive accuracy. Hyperparameter tuning will be applied to optimize their performance. Models will be train on NVIDIA GPU using library cuDF and cuML through raipidsAI, combined with other basic library such as NumPy, Pandas, and Scikit-Learn.

4.1 Feature Engineering

We will use anomaly detection to identify outliers in the accepted loans. Since rejected loans may exhibit higher default risk, we hypothesize that anomaly scores for accepted loans could highlight high-risk profiles. To compute these scores, we apply the Isolation Forest algorithm, training it on the accepted loans and inferring scores for both accepted and rejected loans. Rejected loans are expected to have higher anomaly scores.

We then normalize these scores by dividing each by the maximum anomaly score across both datasets. This normalized value becomes a new feature, indicating how closely an accepted loan resembles high-risk profiles, thereby improving the model's ability to predict loan defaults.

4.2 Logistic Regression & Weighted Logistic Regression

Logistic Regression is a model for binary classification problems, predicting the probability that a given input belongs to a certain class. The basic logistic regression model takes the form:

$$P(y = 1|X) = \frac{1}{1 + e^{-(\beta^\top X)}}$$

For standard weighted linear regression, which is used to address the significant class imbalance, we have one twist to the binary cross entropy loss function, given by

$$\mathcal{L} = -\sum_{i=1}^{n} \frac{N}{n_{\text{classes}} N_c} \left[y_i \log(P(y_i)) + (1 - y_i) \log(1 - P(y_i)) \right]$$

where n_{classes} is the number of classes, N_c is the number of samples in the class to which the data point i belongs. Such weights ensure that minority classes have a more significant impact during model training to mitigate the effect of class imbalance.

4.3 Random Forest

Random Forest is an ensemble learning method designed for classification tasks. It constructs multiple decision trees during training and aggregates their outputs to improve accuracy and reduce overfitting. Each tree is trained on a bootstrap sample, which is a random sample drawn with replacement from the original dataset. If the training set has N samples, each tree is trained on a bootstrap sample $\mathbf{X}^k \subseteq \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $N_k = N$ but some samples may be repeated while others are excluded. This randomness helps generate diverse trees, which collectively improve generalization performance.

For each decision tree, the key operation is determining the best split at each node. Rather than considering all features, Random Forest randomly selects a subset of m features (where $m \ll d$) and chooses the best one based on a splitting criterion. The most commonly used criterion is the *Gini Impurity* for classification, which for a node with class proportions p_i is computed as:

$$I_{\text{Gini}} = 1 - \sum_{i=1}^{C} p_i^2$$

where C is the number of classes, and p_i is the proportion of samples from class i in the node. The feature that minimizes the Gini impurity across all possible splits is chosen to partition the data at that node.

In classification, the final prediction is obtained by majority voting across all the individual trees. For a new sample $\mathbf{x}$, the predicted class $\hat{y}$ is determined by:

$$\hat{y} = \text{mode}\left(\{T_k(\mathbf{x})\}_{k=1}^K\right)$$

where $T_k(\mathbf{x})$ represents the prediction of the k-th tree, and K is the total number of trees in the forest.

4.4 Gradient Boosting

Gradient Boosting is an ensemble learning technique that builds a strong classifier by combining multiple weak learners, in this case decision trees. It involves iteratively training trees, where each new tree attempts to correct the errors made by the previous ones. Gradient boosting adds trees sequentially, with each one trained to minimize the loss function by focusing on the residuals (errors) of the existing model. The final model is the sum of all the individual trees, with each tree's contribution controlled by a learning rate parameter. Gradient boosting is highly effective for both classification and regression tasks, and its flexibility in tuning hyperparameters such as the number of trees, depth, and learning rate makes it adaptable to various types of data.

5 Results

We evaluated the performance of four machine learning models—Logistic Regression, Weighted Logistic Regression, Random Forest, and Gradient Boosting—on the task of predicting whether a loan will be fully paid or charged off. To ensure robust evaluation, we split the dataset into training (80%) and testing (20%) sets. For Gradient Boosting, we used 10% of the training set as a validation set for early stopping. For Random Forest, we similarly split the training data, using 20% of it as a validation set for hyperparameter tuning, as early stopping is not applicable to this model.

Based on Figure 1, the distribution of the anomaly scores for the accepted and rejected data appears to be quite similar. This similarity suggests that the anomaly score feature may not provide significant discriminatory power between these two classes. Consequently, we anticipate that incorporating this feature into the model is unlikely to lead to a substantial improvement in its performance.

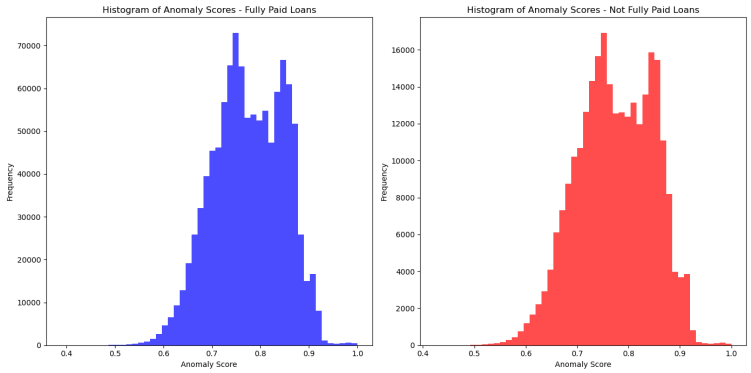


Figure 1: Distribution of the anomaly score feature

Table 1 summarize the accuracies for logistic regression models. We utilized the Logistic Regression algorithm provided by the RAPIDS cuML library for GPU-accelerated machine learning tasks. The accuracy results can be summarized below

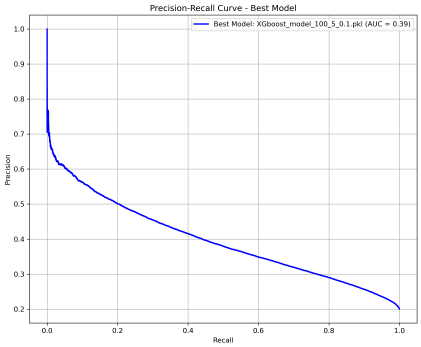
Model	**Accuracy**
Logistic Regression (with Anomaly Score)	0.7996
Weighted Logistic Regression (with Anomaly Score)	0.6761
Logistic Regression	0.7996
Weighted Logistic Model	0.5945

Table 1: Logistic Models Accuracy Results

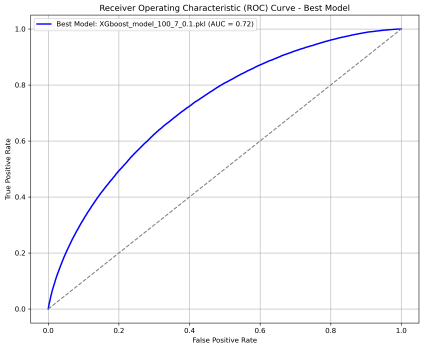
We know that for an imbalanced dataset, accuracy is not the best metric to justify the quality of the model, hence we now look at the ROC Curve and the Precision-Recall (PR) Curve. We found that for both curves, with or without anomaly score, the standard logistic regression model performs better than the weighted logistic regression when comparing the AUC, and the model has slightly higher AUC given the anomaly score. The AUC of 0.35 and precision of 0.8 indicates that most of the accuracy comes from correctly classifying the majority class, while the low AUC for PR curve indicates that the model struggles to predict accurately for the positive class, ie. people who doensn't pay the loan on time.

We fine-tuned the Random Forest model using the validation set, selecting the best model from 36 hyperparameter combinations based on both ROC AUC and Precision-Recall AUC. Given the class imbalance in our dataset, we prioritized the Precision-Recall AUC, as it is more informative for evaluating model performance in such scenarios. After tuning, we selected two sets of hyperparameters. Both models use 250 estimators, a maximum depth of 10 `max_features` set to the square root of the total number of features, and minimum samples leaf of 3. The model with the anomaly score feature uses a minimum samples split of 6, while the model without it uses 4. After testing, we found no significant differences in performance between the model with the anomaly score feature and the one without it, based on both ROC AUC and Precision-Recall AUC.

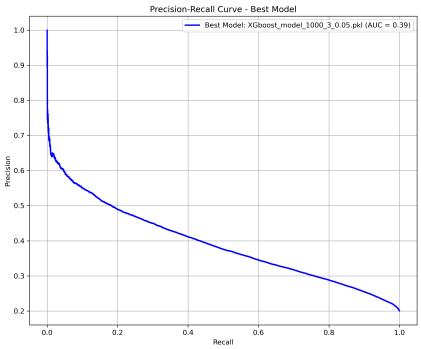
For Gradient Boosting, we fitted 36 different combinations of different hyperparameters for number of estimators, maximum depth of weak learners, and learning rate. The result for accuracy can be obtained at Table 3. It is interesting to observe that although we obtained approxiamtely the same results for different metrics for XGB models with and without anomaly score, when using anomaly score, we were able to obtain approximately the same result using 100 estimators, while without anomaly score we needed 1000. This means that we were able to obtain good results using only 10% of the estimators if we include the anomaly score.



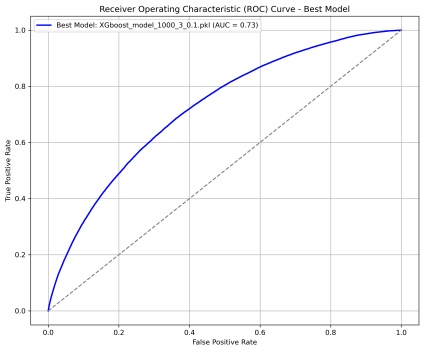
(a) PR Curve for XGB model with Anomaly Score (AUC = 0.39)



(b) ROC Curve for XGB model with Anomaly Score (AUC = 0.72)



(c) PR Curve for XGB Logistic model (AUC = 0.39)



(d) ROC Curve for XGB Logistic model (AUC = 0.73)

Figure 2: Comparison of Precision-Recall and ROC Curves for Different Models

Model	**Accuracy**	**ROC AUC**	**Precision-Recall AUC**
Random Forest (with Anomaly Score)	0.8034	0.71	0.37
Gradient Boosting (with Anomaly Score)	0.8027	0.73	0.372
Random Forest	0.8022	0.71	0.4302
Gradient Boosting	0.8045	0.73	0.40

Table 2: Ensemble Models Accuracy Results

6 Conclusion

This study demonstrated the efficacy of machine learning models, particularly Gradient Boosting, in credit risk prediction. While traditional logistic regression offered a baseline, ensemble methods like Random Forest and Gradient Boosting achieved superior results by addressing complex feature interactions and class imbalances. The anomaly score feature provided a unique perspective, highlighting its potential for improving efficiency but showing limited impact on accuracy.

The findings suggest that future efforts should focus on enhancing feature engineering techniques and exploring additional data augmentation methods to improve model performance for minority classes.

7 Contribution

All group members have submitted the course evaluations.

- Timothy: responsible for data engineering and random forest models.
- Son Nguyen: Responsible for logistic regression models and gradient boosting models.

8 Appendix

n_estimators	max_depth	learning_rate	accuracy
100	3	0.01	0.7997
100	3	0.1	0.8029
100	3	0.05	0.8018
100	5	0.01	0.7997
100	5	0.1	0.8036
100	5	0.05	0.8027
100	7	0.01	0.7997
100	7	0.1	0.8037
100	7	0.05	0.8032
100	9	0.01	0.7998
100	9	0.1	0.8040
100	9	0.05	0.8037
500	3	0.01	0.8016
500	3	0.1	0.8041
500	3	0.05	0.8036
500	5	0.01	0.8028
500	5	0.1	0.8041
500	5	0.05	0.8041
500	7	0.01	0.8035
500	7	0.1	0.8042
500	7	0.05	0.8043
500	9	0.01	0.8038
500	9	0.1	0.8033
500	9	0.05	0.8043
1000	3	0.01	0.8028
1000	3	0.1	0.8042
1000	3	0.05	0.8042
1000	5	0.01	0.8035
1000	5	0.1	0.8043
1000	5	0.05	0.8045
1000	7	0.01	0.8040
1000	7	0.1	0.8030
1000	7	0.05	0.8044
1000	9	0.01	0.8042
1000	9	0.1	0.8011
1000	9	0.05	0.8039

Table 3: Accuracy Results for Different Hyperparameters combinations for XGBoost model, without anomaly score

References

[1] Yao-Chi Chang, Kai-Hsiang Chang, and Gwo-Jen Wu. Application of extreme gradient boosting trees in the construction of credit risk assessment models for financial institutions. *Applied Soft Computing*, 73:914–920, 2018.

[2] Xia Li. Predicting-default-clients-of-lending-club-loans. `https://github.com/yanxiali/Predicting-Default-Clients-of-Lending-Club-Loans`, 2023. Accessed: 2024-11-29.

[3] Zheng Tian, Jiawei Xiao, Hao Feng, and Yong Wei. Credit risk assessment based on gradient boosting decision tree. *Procedia Computer Science*, 174:150–160, 2020.