Predicting Credit Default: A Machine Learning Approach to the "Give Me Some Credit" Kaggle Challenge

Timofey Davidyuk

September 4, 2025

Abstract

This paper details the end-to-end process of developing a machine learning model to predict the probability of consumer credit default. The work was conducted as part of the Kaggle competition "Give Me Some Credit". The dataset, comprising 150,000 anonymized credit records, presents a classic highly imbalanced binary classification problem. We performed extensive exploratory data analysis (EDA), addressing issues such as missing data, skewed distributions, and the presence of anomalous "magic values". A key part of our methodology was a structured model pipeline that efficiently compared multiple algorithms, including Logistic Regression, Random Forest, and Gradient Boosting methods. Feature engineering, including the creation of late-payment flags and debt-to-income ratios, proved crucial. Ultimately, Gradient Boosting models without synthetic oversampling (SMOTE) performed best, achieving an AUC score of over 0.86 on the held-out test set. Our findings suggest that careful data preprocessing and the use of powerful ensemble methods are highly effective for credit risk modeling.

1 Introduction

Credit risk modeling is a critical task for financial institutions, aiming to assess the likelihood that a borrower will default on their debt. Accurate models enable better lending decisions, risk management, and economic stability. The "Give Me Some Credit" Kaggle competition provides a realistic dataset for tackling this problem, challenging participants to predict whether someone will experience serious financial distress within two years.

This report documents our machine learning pipeline for this competition, covering data exploration, preprocessing, feature engineering, model selection, and results. The associated code is available on GitHub: `https://github.com/weaknessofuniverse/credit-scoring-ml`.

2 Data Description and Exploration

The dataset contains 150,000 training samples with 11 features, one of which is the binary target variable, `SeriousDlqin2yrs`. The features include demographic information (age,

number of dependents) and financial history variables (number of past due payments, debt ratio, monthly income).

2.1 Initial Analysis

Our initial analysis revealed two primary challenges:

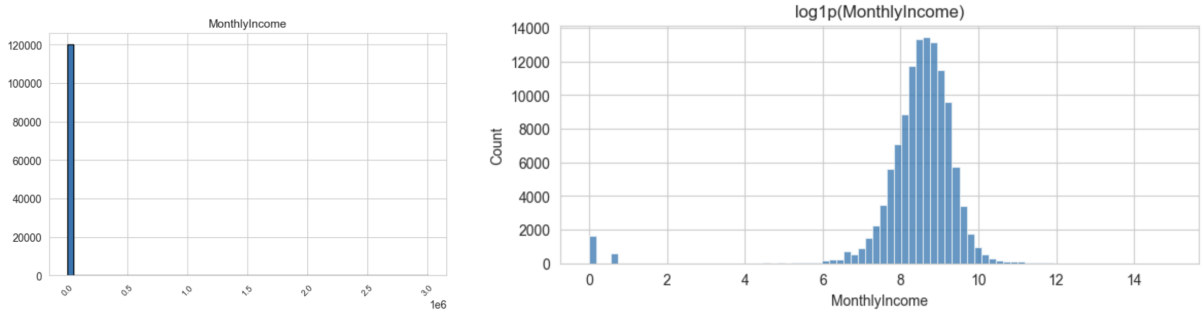
- **Class Imbalance**: The target variable is highly imbalanced, with only 6.68% of customers labeled as having experienced financial distress (positive class).
- **Data Quality Issues**: We identified significant missingness, particularly in the `MonthlyIncome` field (almost 20% missing). Furthermore, the late-payment fields (`NumberOfTime30-59DaysPastDueNotWorse`, etc.) contained anomalous values of 96 and 98, which we treated as missing data (likely placeholders for "unknown").

2.2 Handling Missing and Anomalous Data

The "magic values" (96, 98) in the delinquency fields were first replaced with `NaN`. Subsequently, all missing values in these fields were imputed with 0, under the assumption that a missing delinquency record indicates no history of late payment. `MonthlyIncome` was imputed using the median value from the training set, and `NumberOfDependents` was imputed using the mode.

2.3 Feature Distributions and Transformations

Many features exhibited highly skewed distributions. `MonthlyIncome`, for instance, had a very long right tail. To handle this, we applied a log transformation ($log(1 + x)$) and clipped extreme values at the 99th percentile to prevent them from overly influencing the model. Variables like `RevolvingUtilizationOfUnsecuredLines` and `DebtRatio` were also clipped.



(a) Original `MonthlyIncome` (b) Log-transformed `MonthlyIncome`

Figure 1: Effect of log transformation on the distribution of `MonthlyIncome`.

3 Feature Engineering

To enhance predictive power, we created several new features:

- **Age Groups**: Binned age into categorical groups (18-25, 26-35, ..., 76+).

- **Delinquency Flags**: Created binary features indicating whether a customer had *any* history of a specific type of late payment (e.g., `Has_NumberOfTimes90DaysLate`).
- **Total Late Payments**: A summed score of all late payments across different severity levels.
- **Income Per Dependent**: `MonthlyIncome / (NumberOfDependents + 1)`.
- **Debt-to-Income Ratio**: `DebtRatio * MonthlyIncome`, providing an absolute measure of debt burden.

4 Modeling Approach

We employed a three-stage modeling strategy to identify the best performer efficiently.

4.1 Model Descriptions

We selected a diverse set of algorithms known for their effectiveness in tabular data problems to ensure a robust comparison.

- **Logistic Regression (LR)**: A linear model that estimates the probability of the default class. Its simplicity, interpretability, and strong performance on linearly separable problems make it a reliable baseline.
- **Random Forest (RF)**: An ensemble learning method that constructs a multitude of decision trees during training. Each tree votes on the outcome, and the class with the most votes becomes the model's prediction. RF is robust to overfitting and effective at capturing non-linear relationships.
- **Gradient Boosting (GB)**: Another powerful ensemble technique that builds trees sequentially. Each new tree is trained to correct the errors made by the previous ones. This iterative approach often leads to high predictive accuracy but can be more prone to overfitting than Random Forest if not properly regularized.
- **Histogram-Based Gradient Boosting (HistGB)**: A highly optimized variant of Gradient Boosting from scikit-learn. It bins continuous feature values into histograms, which dramatically speeds up the training process. It is often the best performer on large datasets with continuous features.
- **Support Vector Machine (SVM)**: A model that finds the optimal hyperplane to separate classes in a high-dimensional space. We used a linear kernel for efficiency. While powerful, it often struggles with large datasets and requires careful scaling of features.
- **K-Nearest Neighbors (KNN)**: A simple, instance-based learning algorithm that classifies a data point based on how its neighbors are classified. Its performance is highly dependent on the distance metric and the value of K, and it is computationally expensive during prediction.

4.2 Addressing Class Imbalance

The severe class imbalance (6.7% vs. 93.3%) is a central challenge. We employed two strategies:

1. **Class Weighting**: For models that support it (e.g., LR, RF, SVM), we set `class_weight='balanced'`. This instructs the algorithm to penalize misclassifications of the minority class more heavily than those of the majority class.

2. **Synthetic Minority Oversampling (SMOTE)**: We tested the SMOTE algorithm, which generates synthetic examples of the minority class in the feature space. This creates a more balanced dataset before training. We compared the performance of all models with and without SMOTE.

4.3 The SMOTE Technique

A central challenge in credit scoring is the severe class imbalance, where the number of non-defaulters vastly outnumbers the defaulters. To mitigate this, we employed and evaluated Synthetic Minority Over-sampling Technique (SMOTE)

Unlike naive random oversampling, which simply duplicates existing minority class instances, SMOTE is a more sophisticated algorithm that generates synthetic examples. It operates in feature space by interpolating between existing minority instances. For a given instance from the minority class, SMOTE:

1. Finds its k-nearest neighbors (typically $k = 5$).

2. Randomly selects one of these neighbors.

3. Creates a new, synthetic data point at a randomly selected point on the line segment connecting the original instance and its selected neighbor.

This approach effectively "fills in" the feature space for the underrepresented class, encouraging the learning algorithm to create more generalized and less overfit decision boundaries.

However, SMOTE is not a panacea. Its effectiveness is highly dependent on the dataset. It can sometimes introduce noise if the synthetic instances are generated in regions overlapping with the majority class or if the feature space is not densely populated. In our experimentation, we rigorously compared the performance of all top models with and without SMOTE-applied during cross-validation to prevent data leakage. Interestingly, as detailed in the results (Table X), the use of SMOTE consistently led to a decrease in model performance for this particular dataset. This suggests that the tree-based ensemble methods were sufficiently robust to the inherent imbalance, potentially due to their internal weighting mechanisms, and that the synthetic samples may not have accurately captured the complex, real-world dynamics of credit default.

4.4 Stage 1: Quick Scan

We evaluated seven different classifiers (Logistic Regression, Random Forest, Gradient Boosting, HistGradientBoosting, SVM, Bagging, KNN) using a randomized search on a 30% subset of the data. This quickly revealed that tree-based ensemble methods (Gradient Boosting and HistGradientBoosting) were the most promising.

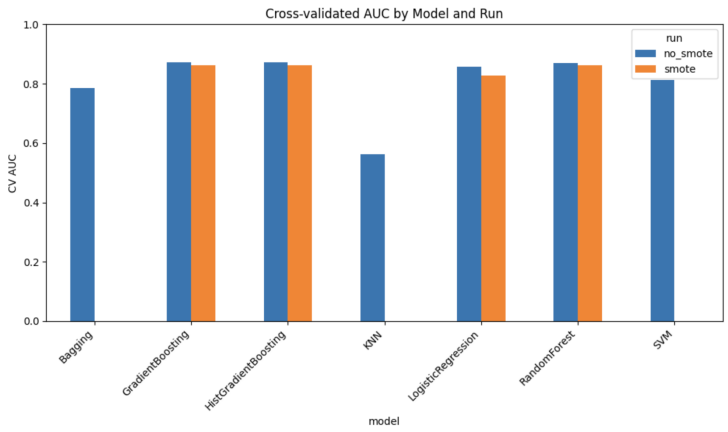


Figure 2: AUC scores for models

4.5 Stage 2: Refinement

The top four models from Stage 1 were retrained on the full dataset with more hyperparameter combinations. We also evaluated the effectiveness of SMOTE for handling class imbalance. Results showed that tree-based models (GB, HistGB, RF) performed best *without* SMOTE.

Table 1: Cross-Validation AUC scores for top models after refinement.

Model	Mode	CV AUC
GradientBoosting	no_smote	0.8710
HistGradientBoosting	no_smote	0.8710
RandomForest	no_smote	0.8684
HistGradientBoosting	smote	0.8628
GradientBoosting	smote	0.8626

4.6 Stage 3: Final Training

The two best models, Gradient Boosting and HistGradientBoosting, underwent a final round of hyperparameter tuning using a more exhaustive grid search. The best parameters from this search were used to train the final models saved for submission.

5 Results and Discussion

The final models were used to generate predictions on the official Kaggle test set. The results are shown in the table below. GradientBoosting without SMOTE achieved the highest private score.

5.1 Key Findings

- Gradient Boosting methods were the top performers for this task, effectively capturing the complex relationships in the data.

Table 2: Final Kaggle submission results.

Submission	**Public Score**	**Private Score**
GradientBoosting (no SMOTE)	**0.85944**	**0.86500**
HistGradientBoosting (no SMOTE)	0.85964	0.86387
GradientBoosting (SMOTE)	0.85298	0.85897
HistGradientBoosting (SMOTE)	0.85237	0.85851

- Contrary to common practice for imbalanced datasets, SMOTE did not improve performance. This suggests that the tree-based algorithms themselves are robust to the imbalance, or that the implicit cost-sensitive learning from using class_weight='balanced' was sufficient.
- Feature engineering, particularly the creation of delinquency flags and the debt-to-income ratio, provided valuable signals to the models.

6 Conclusion

This project successfully demonstrated a complete pipeline for building a credit risk model. Through meticulous EDA, thoughtful preprocessing, and strategic feature engineering, we prepared the data for modeling. A structured approach to model selection and hyperparameter tuning identified HistGradientBoosting as the best algorithm for this specific problem. The final model achieves a high level of accuracy in predicting credit default, as validated by its performance in the Kaggle competition. Future work could explore more sophisticated feature engineering, different sampling techniques, or model stacking to potentially squeeze out further performance gains.