Credit Default Risk Forecast & Credit Decision

Yankai Zhao
Department of Electrical & Computer Engineering
Lehigh University
Bethlehem, PA, USA
yaz624@lehigh.edu

***Abstract*—Credit risk management is critical for financial institutions, centered on the Risk–Return Trade-off: maximizing income while minimizing losses from defaults. This paper presents a comparative study of machine learning models for predicting credit card defaults, aligning model outputs with business P&L objectives.**

We utilize the UCI Credit Card dataset [1] to evaluate and analyze three tree-based algorithms—Decision Trees, Random Forests (*Bagging*), and XGBoost (*Gradient Boosting*)—focusing on their distinct underlying mechanisms. Recognizing the asymmetric cost of misclassification in banking, we implement a Neyman–Pearson-style, cost-sensitive thresholding strategy to strictly control the False Positive Rate (FPR) while maximizing recall.

Experimental results, benchmarked by AUC and KS statistics, show that hyperparameter-tuned XGBoost achieves the best discrimination (validation AUC ≈ 0.78). XGBoost offers a more favorable FPR–Recall trade-off than baseline models, providing a practical framework for identifying high-risk customers and maintaining a healthy loan portfolio.

***Index Terms*—Credit Risk, Credit Cards, Decision Trees, Random Forest, XGBoost, Bagging, Boosting, Neyman–Pearson**

I. Introduction

How does a bank make money? A simplified Profit & Loss (P&L) view of a credit card portfolio is

$$P \approx II + F - FC - CL - OE \tag{1}$$

- P = Profit
- II = Interest Income
- F = Fees
- FC = Funding Cost
- CL = Credit Losses
- OE = Operating Expenses

While revenues come from interest (APR) and various fees (annual fee, interchange, late fees, etc.), the most volatile component is *Credit Losses*—money lost when a customer defaults.

This creates a classic **risk–return trade-off**: issuing more credit cards and higher limits increases potential interest and fee income, but also elevates the probability and severity of default. Therefore, the goal of a credit scoring model is not merely maximizing classification accuracy, but improving the portfolio-level P&L. In particular, the bank faces an asymmetric cost structure:

- **False Positive (Type I Error)**: rejecting a truly good customer (non-default). Cost: lost interest and fee income, possible customer churn.
- **False Negative (Type II Error)**: approving a truly bad customer (default). Cost: loss of principal and interest, collection expenses, and higher capital/provision requirements.

This paper explores how modern tree-based models—Decision Trees, Random Forests, and XGBoost—can be applied to a real-world credit card dataset to manage this trade-off. Beyond model comparison, we explicitly study how different decision thresholds impact FPR and recall, and propose a Neyman–Pearson-style thresholding rule that prioritizes a strict cap on FPR while preserving as much recall as possible.

II. Dataset Description and Preprocessing

A. UCI Default of Credit Card Clients Dataset

We use the public *Default of Credit Card Clients* dataset from the UCI Machine Learning Repository [1]. The dataset consists of 30,000 credit card clients from Taiwan, with information collected from April to September 2005. Each record corresponds to a single customer and includes:

- **ID**: client identifier (not used as a predictor).
- **LIMIT_BAL**: amount of given credit in NT dollars.
- **SEX**: gender (1 = male, 2 = female).
- **EDUCATION**: education level (1 = graduate school, 2 = university, 3 = high school, 4 = others).
- **MARRIAGE**: marital status (1 = married, 2 = single, 3 = others).
- **AGE**: age in years.
- **PAY_0–PAY_6**: repayment status in the past six months (-1 = pay duly, 0 = use of revolving credit, 1–8 = months of delay).
- **BILL_AMT1–BILL_AMT6**: amount of bill statement (NT dollar) from April to September 2005.
- **PAY_AMT1–PAY_AMT6**: amount of previous payment (NT dollar) from April to September 2005.
- **default_payment_next_month**: target label (1 = default, 0 = non-default).

The target variable is moderately imbalanced: 6,636 out of 30,000 customers (22.1%) default in the next month, while 23,364 (77.9%) do not default. This level of imbalance is realistic for retail credit card portfolios and justifies the use of AUC, KS, and cost-sensitive analysis rather than raw accuracy.

B. Data Quality and Cleaning

We first verify data completeness and class distribution. The original dataset contains no missing values in any fea-

ture, which avoids the need for imputation.However, several categorical variables use undocumented codes (e.g., EDUCATION values 0, 5, 6 and MARRIAGE value 0). Following common practice, we group rare or undefined categories into the "others" class to stabilize model estimates [2]. Concretely:

- `EDUCATION` values $\{0, 5, 6\}$ are recoded as 4 (others).
- `MARRIAGE` value 0 is recoded as 3 (others).

For tree-based models, monotonic transformations and scaling of numeric features do not change the split structure. Therefore, we keep the original monetary units (NT dollars) for `LIMIT_BAL`, `BILL_AMT`, and `PAY_AMT`. Categorical variables are encoded as integers as in the original dataset.

C. Exploratory Data Analysis

Exploratory analysis reveals several important patterns:

- **Credit Limit Distribution**: The most common limits are 50,000, 30,000 and 20,000 NT dollars. Defaults are more concentrated in low-to-medium credit limit ranges, while very high limits are associated with lower default rates.
- **Payment Behavior**: The repayment status features (`PAY_0`–`PAY_6`) carry strong information. Customers with recent delays (e.g., `PAY_0` ≥ 1) have a significantly higher default rate in the next month.
- **Correlation Structure**: Bill amounts (`BILL_AMT1`–`BILL_AMT6`) are **highly correlated** (reflecting temporal persistence of outstanding balances), whereas payment amounts (`PAY_AMT1`–`PAY_AMT6`) show weaker but noticeable correlations. This naturally motivates the use of tree ensembles, which can effectively handle such correlated predictors.
- **Demographics**: Age, education, and marital status show weaker but non-negligible relationships with default, mainly through interactions with payment behavior.

D. Train–Validation Split

We split the dataset into an 80% training set and a 20% validation set using stratified sampling to preserve the default rate in both splits. For models that require threshold tuning (Section IV), the training set is further split into an internal training fold and a calibration fold via `train_test_split` with stratification on the target label.

III. Modeling Framework

We focus on three tree-based models with increasing complexity: a single Decision Tree, a Random Forest, and XGBoost. All models use the same set of 23 predictor variables:

$$\mathcal{P} = \left\{ \begin{array}{c} \texttt{LIMIT\_BAL}, \texttt{SEX}, \texttt{EDUCATION}, \texttt{MARRIAGE}, \texttt{AGE}, \\ \texttt{PAY\_0}, \ldots, \texttt{PAY\_6}, \\ \texttt{BILL\_AMT1}, \ldots, \texttt{BILL\_AMT6}, \\ \texttt{PAY\_AMT1}, \ldots, \texttt{PAY\_AMT6} \end{array} \right\}$$

A. Decision Tree (CART)

The baseline model is a binary Decision Tree using the CART algorithm. For classification, CART selects splits that maximize impurity reduction, measured by Gini impurity or entropy:

$$\text{Gini}(S) = 1 - \sum_k p_k^2, \quad \text{Entropy}(S) = -\sum_k p_k \log p_k, \quad (2)$$

where p_k is the class proportion in node S. The split producing the largest information gain is chosen recursively until stopping criteria are met.

We perform grid search over maximum depth, minimum samples per split, minimum samples per leaf, and the impurity criterion. Cross-validation on the training set selects a relatively shallow tree (e.g., depth 6–8), which balances interpretability and variance. Visualization of the learned tree shows intuitive splits: recent payment status (`PAY_0`, `PAY_2`) and bill amounts are dominant at the top levels.

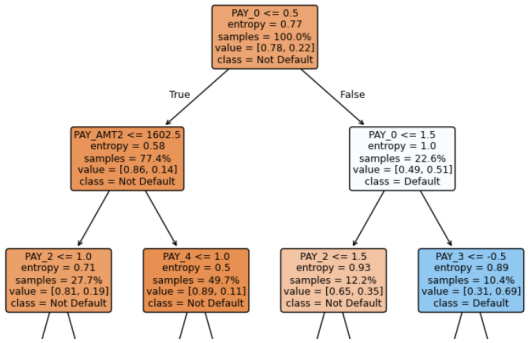

Fig. 1. Top levels of the learned Decision Tree. The root node splits on recent repayment status `PAY_0`, followed by nodes on `PAY_4`, `PAY_2`, `PAY_3`,`PAY_AMT2`. Customers with recent delinquencies and large outstanding balances are pushed to the right-hand "Default" leaves, while regular payers with larger recent payments move to the "Non-Default" leaves.

Figure 1 highlights that the tree recovers intuitive business rules: recent delinquency status is the primary driver, and conditional on that, both current bill amount and recent repayment level further separate high-risk and low-risk groups. From a risk-management perspective, a key advantage of a single Decision Tree is its **interpretability**. Each path from the root to a leaf corresponds to a human-readable rule such as

"PAY_0 > 1.5 *& PAY_AMT3* $< 12150 \Rightarrow$ *high default risk"*.

Such rules can be directly reviewed by credit analysts and aligned with existing policies. In this project, the upper levels of the tree essentially recover intuitive business logic: recent delinquency status determines whether a customer is screened as potentially risky, and conditional on that, the current bill amount and recent repayment amounts refine the segmentation into higher- and lower-risk groups.

B. Random Forest

Random Forest extends decision trees via *bagging*. With B trees, each tree is trained on a bootstrap sample and uses

random feature subsampling at each split:

$$\hat{f}_{\mathrm{RF}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(x). \tag{3}$$

This averaging reduces variance and improves generalization, especially in the presence of noisy or correlated features.

We tune the number of trees, maximum depth, and the number of features considered at each split. Feature importance analysis for the Random Forest confirms that repayment status, recent bill amounts, and credit limit are the most influential predictors, aligning with domain intuition.

C. XGBoost

XGBoost (eXtreme Gradient Boosting) is a powerful boosting method that builds trees sequentially. Each new tree f_t is trained to approximate the negative gradient of a differentiable loss function, such as logistic loss:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(x_i), \tag{4}$$

where η is the learning rate. XGBoost includes regularization (on tree depth and leaf weights), shrinkage, and subsampling, which help prevent overfitting.

We construct DMatrix objects for train and validation sets and tune hyper-parameters such as maximum tree depth, learning rate, number of estimators (boosting rounds), subsample ratio, and column subsampling. Early stopping on validation AUC selects the best iteration. In our experiments, XGBoost achieves a validation AUC of about 0.78 at around 450 boosting rounds, substantially outperforming simpler tree models.

Although gradient boosting ensembles are often viewed as "black boxes", we improve the interpretability of the XGBoost model by computing **SHAP (SHapley Additive exPlanations)** values.

Figure 2 shows a local SHAP "waterfall" plot for a single customer. The horizontal axis is the log-odds score $f(x)$. Starting from the average score $E[f(X)]$ at the bottom, each bar shows how one feature pushes this customer's score up or down. Blue bars (e.g., PAY_0 and BILL_AMT3) decrease the default score and therefore protect the customer from being classified as high risk, while red bars (e.g., BILL_AMT2 and BILL_AMT4) increase the score and push the prediction towards default. This gives a case-level explanation of *why* the model judged this particular customer as relatively safe.

To obtain a global view, we also use SHAP summary (beeswarm) plots, as illustrated in Figure **??**. Each row corresponds to a feature, each dot to one customer, and the position on the horizontal axis is the SHAP value (impact on the default score). The color encodes the feature value (red = high, blue = low). The plot shows that recent delinquency status (PAY_0, PAY_2), current bill amounts (BILL_AMT1–2), and recent payments (PAY_AMT1) are consistently the most influential drivers of the model, and that higher PAY_0 or larger bill amounts tend to push customers towards default. In this way, SHAP bridges the gap between predictive performance and transparency, allowing risk analysts to verify that the boosted model is learning economically meaningful patterns rather than spurious correlations.

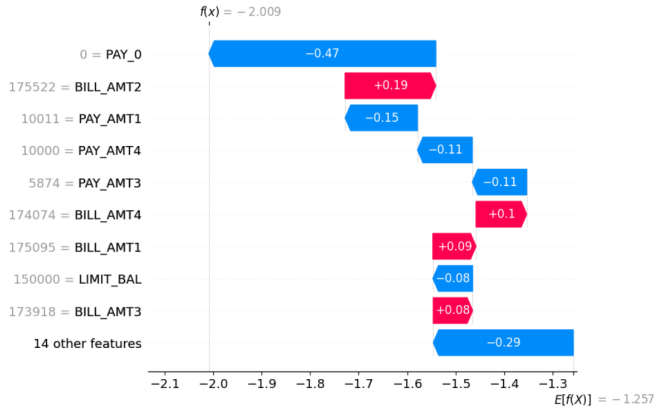

Fig. 2. Local SHAP explanation for one customer. Blue bars decrease the default score (safer), while red bars increase it (riskier), starting from the portfolio average score $E[f(X)]$.

D. Other Baseline Models

For completeness, we also experimented with AdaBoost and CatBoost classifiers on the same feature set. These models achieve AUC values comparable to or slightly below Random Forest. Given space constraints and the superior performance of XGBoost, we focus the detailed analysis on Decision Tree, Random Forest, and XGBoost.

IV. Risk Control Strategy

Standard classifiers typically use a default decision threshold of 0.5 on the predicted probability of default, implicitly treating False Positives (FP) and False Negatives (FN) as equally costly. In credit card lending, this is unrealistic: approving bad customers (FN) leads to direct credit losses, while rejecting good customers (FP) primarily hurts revenue and customer experience.

To incorporate this asymmetry, we adopt a **cost-sensitive thresholding** strategy inspired by the Neyman–Pearson paradigm. Let

$$\mathrm{FPR} = \frac{FP}{FP+TN}, \qquad \mathrm{Recall} = \frac{TP}{TP+FN}. \tag{5}$$

The bank may specify a hard upper bound α on FPR (e.g., 1–5%), reflecting how many good customers it is willing to mistakenly reject. Under this constraint, we seek the threshold that maximizes recall among defaulters.

Operationally, we proceed as follows:

1) Split the validation data into a *calibration* subset and an *evaluation* subset using stratified sampling.
2) On the calibration subset, compute predicted scores s_i and sort unique thresholds t from high to low.
3) For each threshold t, compute empirical FPR and recall on the calibration data:

$$\widehat{\mathrm{FPR}}(t) = \frac{FP(t)}{FP(t)+TN(t)}, \quad \widehat{\mathrm{Recall}}(t) = \frac{TP(t)}{TP(t)+FN(t)}.$$

4) Among all thresholds with $\widehat{\mathrm{FPR}}(t) \leq \alpha$, select the one with the largest $\widehat{\mathrm{Recall}}(t)$. This uses the empirical distribution of scores, especially for non-default customers, to approximate the Neyman–Pearson optimal threshold.
5) Finally, report realized FPR and recall on the hold-out evaluation subset under this selected threshold, together with bootstrap confidence intervals for recall.

This procedure allows us to "lock in" a desired maximum business false rejection rate (FPR) while quantifying how many defaulters can be detected under such a conservative requirement.

V. Experimental Results

A. Evaluation Metrics

We evaluate the models using standard industry metrics:

- **ROC AUC**: area under the receiver operating characteristic curve, measuring global ranking quality of the score.
- **KS Statistic**: the maximum vertical distance between the empirical CDFs of scores for non-defaulters and defaulters; higher KS indicates better separation between the two populations.
- **FPR and Recall**: computed at operating points selected by the NP-style thresholding procedure.

B. Model Comparison

Table I summarizes the AUC and KS values on the validation set for the three main models.

TABLE I
Model Performance on Validation Set

Model	ROC AUC	KS (approx.)
Decision Tree	≈ 0.65	moderate
Random Forest	≈ 0.66	slightly higher
XGBoost	≈ 0.78	highest separation

The single Decision Tree provides a useful baseline but suffers from limited discrimination. Random Forest slightly improves AUC and KS by reducing variance through bagging. XGBoost delivers a substantial jump in AUC and KS, indicating that boosting captures more subtle patterns in repayment behavior and balances.

C. Neyman–Pearson Curve for XGBoost

To further visualize this trade-off, we plot an empirical Neyman–Pearson (NP) curve for the XGBoost model, shown in Fig. 3. Each point corresponds to one candidate threshold evaluated on the validation set.

Reading Fig. 3 from left to right, we see that very conservative operating points with low FPR (around 0.02) achieve limited recall (around 0.2). As FPR increases, recall improves steadily and reaches above 0.6 when FPR is allowed to rise close to 0.23. A risk manager can select an operating point to the left of the dashed line (respecting the FPR limit) that still offers as high recall as possible. This provides a clear, visual way to explain the NP-style thresholding rule to non-technical stakeholders: "we cap the probability of rejecting good customers, and within that cap we choose the threshold that catches the maximum fraction of bad customers."

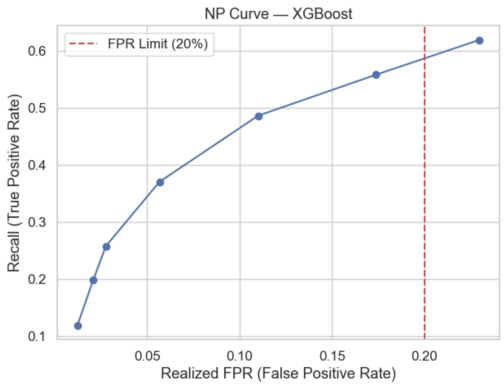

Fig. 3. Empirical NP curve for XGBoost. The horizontal axis is the realized False Positive Rate (FPR), and the vertical axis is recall (True Positive Rate). Each point corresponds to a different decision threshold. The dashed vertical line marks a business constraint on FPR (e.g., 20%).

VI. Conclusion

This project demonstrates how tree-based machine learning models can be applied to credit card default prediction while respecting business risk constraints. Using the UCI credit card dataset, we find that advanced ensemble methods such as Random Forest and especially XGBoost significantly outperform a single Decision Tree in terms of AUC and KS separation.

More importantly, by moving beyond a fixed 0.5 threshold and adopting a cost-sensitive, Neyman–Pearson-style thresholding strategy, we align model evaluation with actual banking P&L: the FPR constraint reflects the tolerance for rejecting good customers, while recall captures the proportion of bad customers we can detect under that constraint. Our experiments show that under very conservative FPR limits, the achievable recall is inherently limited, highlighting the fundamental trade-off imposed by the overlap between good and bad customers in the feature space.

Future work may explore richer feature engineering (e.g., derived utilization ratios and vintage-level behavior), temporal models that capture customer trajectories, and hybrid approaches combining gradient boosting with deep learning for further performance gains.

References

[1] I.-C. Yeh, "Default of credit card clients Dataset," UCI Machine Learning Repository, 2016. [Online]. Available: https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients.
[2] J. Allan, "Cleaning the Credit Card Defaults Dataset," *James Allan's Blog*, 2020. [Online]. Available: https://jmsallan.netlify.app/blog/cleaning-the-credit-card-defaults-dataset/.
[3] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. ACM KDD*, 2016.
[4] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.