Binary Prediction of Credit Card Default Using Machine Learning Algorithms

Yang (Viola) Song, Yian Zha, Xiangyu Liu

{viola.song, anne.zha, swift.liu}@mail.utoronto.ca

***Abstract*—Credit card default prediction is vital for financial institutions to manage risk and ensure profitability. Defaults, such as those contributing to the 2008 financial crisis, highlight the importance of accurate credit risk assessment. This project utilizes the "Default of Credit Card Clients" dataset from the UCI Machine Learning Repository to automate default prediction using machine learning. We evaluate and compare Support Vector Machines (linear, Gaussian, and polynomial kernel), Logistic Regression, and Random Forest to handle the dataset's imbalance and uncover complex patterns. Among these, the Random Forest model outperformed others and achieved an AUC-ROC score of 0.7747 on the test dataset, demonstrating strong discriminatory power. Our results show that machine learning is an effective way to predict credit card defaults, supporting proactive decision-making and improving financial stability.**

***Index Terms*—ECE 1513, Introduction to Machine Learning**

I. Introduction

Predicting credit card default is a critical challenge for financial institutions, as it directly impacts their ability to manage risks and maintain profitability. Defaults occur when borrowers fail to meet their payment obligations, leading to financial strain for both lenders and the broader economy. With the increasing reliance on consumer credit, effective credit risk management has become more important than ever.

Existing solutions for credit card default prediction primarily rely on traditional statistical methods and credit scoring systems. While these approaches provide valuable insights, they often fail to account for complex, non-linear relationships in data, especially when dealing with large, imbalanced datasets. For example, traditional credit scoring models struggle to adapt to rapidly changing customer behaviors or interactions among multiple demographic, financial, and payment features [1]. This limitation leaves room for significant improvements in predictive accuracy and robustness.

In this project, we propose a machine learning-based solution to automate and improve the prediction of credit card defaults. By leveraging supervised learning algorithms such as Logistic Regression, Random Forest, and Support Vector Machines, our approach seeks to uncover complex patterns and relationships in the "Default of Credit Card Clients" dataset whose details are described in Section II. Through careful analysis and performance evaluation, we aim to provide a data-driven tool that enhances credit risk assessment and enables financial institutions to make more informed and proactive decisions. This solution holds the potential to significantly reduce losses, improve customer relationships, and ensure greater financial stability for lenders.

II. Dataset Overview

The "Default of Credit Card Clients" dataset from the UCI Machine Learning Repository contains 30,000 observations of credit card holders in Taiwan from April to September 2005 [2]. It includes 23 variables, such as demographic information, credit history, and financial transactions, with a binary target indicating whether a client defaults on their payment in the following month.

To preprocess the dataset, ambiguous values in the `EDUCATION` and `MARRIAGE` features were removed, and undocumented values in the `PAY_n` columns were rescaled to align with the official description. Correlation analysis revealed that the `PAY_n` features were highly predictive of default, while all `BILL_AMTn` features were redundant due to high inter-correlation. We retained `BILL_AMT1` and dropped the others. The data was then split into training and testing sets for model training. Figure 1 illustrates the correlations with the target.

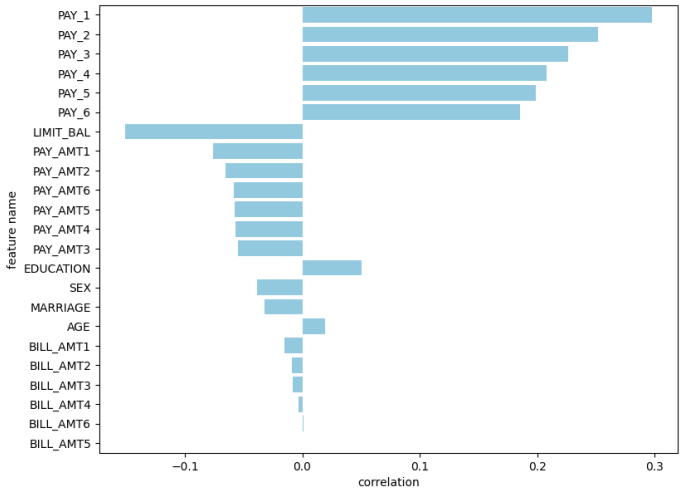

Fig. 1. Correlation analysis between features and the target.

What's more, in our dataset, only 22.12% of the samples are defaulters, while the remaining 77.88% are non-defaulters, indicating a significant class imbalance. This imbalance can cause the model to favor the majority class and leads to biased predictions. To address this, we set

`class_weight='balanced'` in the model, which automatically adjusts the weights of the classes inversely proportional to their frequencies. This increases the penalty for misclassifying the minority class to ensure a fairer learning process. Additionally, we use the ROC-AUC metric for evaluation, which is well-suited for imbalanced datasets, and more detaied will be explained in Section IV.

III. System Design

A. Soft Margin Support Vector Machine

We begin by using Linear SVM due to its simplicity and computational efficiency, particularly when dealing with high-dimensional datasets. Linear SVM provides a robust baseline for understanding linear relationships in the data and is less prone to overfitting compared to kernel-based methods when the dataset is large.

The Hard Margin Support Vector Machine assumes that the training set is perfectly linearly separable, which may not hold true for many real-world datasets. To address this limitation, as shown in Figure 2, the Soft Margin Support Vector Machine allows certain data points to violate the margin constraints by introducing non-negative slack variables, ξ_i, for $i = 1, \ldots, N$. These slack variables quantify how much the constraints are violated for each sample [3].

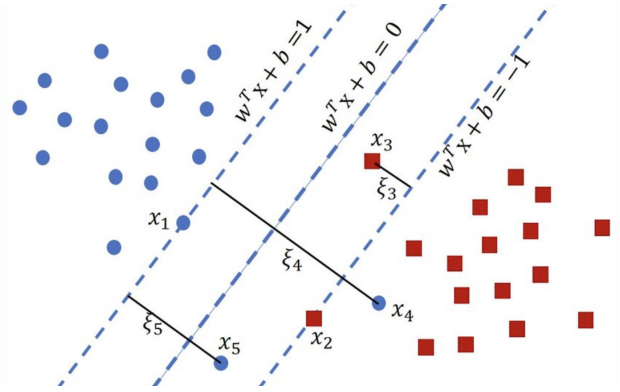

Fig. 2. SVM with soft margin kernel with different cases of slack variables. [4]

The optimization objective for the soft margin formulation balances maximizing the margin and minimizing the classification errors. This is expressed as:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i, \tag{1}$$

subject to the constraints:

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \tag{2}$$

$$\xi_i \geq 0, \quad \forall i, \tag{3}$$

where C is a hyperparameter that controls the trade-off between maximizing the margin and allowing classification errors, with higher values focusing on a wider margin and lower values permitting more violations for smoother boundaries. In the real implementation, we tested `C`: [0.01, 0.1, 1, 100].

B. The Gaussian Kernel (Radial Basis Function)

After using Linear SVM, we may find that a linear decision boundary fails to capture the dataset's complexity due to non-linear relationships or interactions between features. To address this, the Gaussian kernel, or Radial Basis Function (RBF) kernel, offers a more flexible decision boundary by mapping the data into a higher-dimensional space.

The RBF kernel is defined as:

$$K(x, x') = \exp(-\gamma\|x - x'\|^2), \tag{4}$$

where $\|x - x'\|$ represents the Euclidean distance between two data points, while γ determines the kernel's scale or width. A smaller γ results in smoother decision boundaries, whereas a larger γ captures finer, more localized variations. The kernel function evaluates the similarity between data points based on their distance within the feature space [5]. Our tested configurations are `C`: [0.1, 1, 10], and γ: [0.01, 0.1, 1].

C. The Polynomial Kernel

In addition to the RBF kernel, we also utilize the polynomial kernel for SVM. The polynomial kernel maps data into a higher-dimensional space using a polynomial function. It computes the dot product between data points in their original space and their transformed representations in the higher-dimensional space. This kernel is particularly useful for SVM classification tasks involving non-linearly separable data, as the transformation can enable the discovery of a hyperplane that effectively separates different classes [6]. It is defined as:

$$K(x_1, x_2) = \left(x_1^T x_2 + c\right)^d, \tag{5}$$

where x_1 and x_2 are input vectors, c is a constant, and d is the polynomial degree. The degree d determines the model's complexity, with higher values enabling intricate decision boundaries that may overfit, while lower values simplify the model, potentially leading to underfitting. The constant c balances the trade-off between fitting the training data and maintaining a larger margin. A larger c reduces training error but risks overfitting, whereas a smaller c increases training error but encourages a simpler model less sensitive to noise. We tested with `d`: [2, 3, 4], and `c` = 1 in our implementation.

D. Logistic Regression

Logistic regression is an another widely used method for solving binary classification problems. It models the relationship between input features and the probability of an outcome using the sigmoid function. The sigmoid function maps the output of a linear combination of features into a range between 0 and 1, making it particularly effective for scenarios where the data is linearly separable [7]. This probabilistic approach allows logistic regression to not only predict class labels but

also provide confidence scores for the predictions, which can be useful in decision-making processes.

The sigmoid function is expressed as:

$$P(y=1|X) = \frac{1}{1+e^{-(\mathbf{w}^\top \mathbf{X}+b)}} \tag{6}$$

where $\mathbf{w}$ represents the feature weights, $\mathbf{x}$ denotes the feature vector, and b is the bias term. These parameters are learned during training to minimize the difference between the predicted probabilities and the actual labels, typically using optimization algorithms like gradient descent.

The hyperparameters of the logistic regression model were carefully tuned to optimize performance on the imbalanced dataset. Various penalties, such as l_1, l_2, and elastic net, were explored to prevent overfitting and improve generalization. Additionally, the regularization strength C was fine-tuned, with smaller C values providing stronger regularization to manage noise and class imbalance. The tested configuration included `penalty`: [`'l1'`, `'l2'`, `'elasticnet'`], `C`: [0.01, 0.1, 1, 10, 100], and `solver`: [`'lbfgs'`, `'liblinear'`, `'saga'`].

E. Random Forest

Random Forest is an ensemble learning method that improves predictive accuracy and reduces overfitting by combining multiple decision trees. Figure 3 demonstrates this workflow, starting with a dataset used to train several decision trees independently. Each tree is trained on a unique bootstrapped sample of the original dataset, where random sampling with replacement introduces diversity among the trees [8]. This diversity ensures that the ensemble benefits from a wide range of perspectives on the data and makes it more robust and less prone to overfitting compared to individual decision trees.

Each decision tree generates a prediction, labeled in the figure as *Result-1*, *Result-2*, and so on. For classification tasks, these predictions are aggregated through majority voting, where the class predicted most frequently is chosen as the final result. This approach leverages the parallel and independent nature of decision tree training, which effectively reduces variance and improves the model's generalization capability.

For this project, Random Forest was selected due to its inherent strengths in handling imbalanced datasets and its ability to capture complex, non-linear relationships between features and the target variable [10]. Additionally, Random Forest provides high predictive accuracy for binary classification tasks, which makes it a more robust and reliable choice compared to single decision tree models such as CART.

The hyperparameters of the Random Forest model were tuned using grid search. The parameter `n_estimators`, which controls the number of trees in the forest, was tested in the range [50, 100, 200, 500] to balance predictive performance and computational cost. While increasing the number of trees generally enhances accuracy, it also raises computational complexity. The parameter `max_depth`, which defines the maximum depth of each tree, was tuned across [None, 10, 20] to ensure the model captured relevant patterns without overfitting. Additionally, `min_samples_split` was set to [2, 5, 10], and `min_samples_leaf` was set to [1, 2, 4], controlling tree growth and reducing the risk of overfitting [11]. These adjustments helped improve the model's generalization on unseen data.

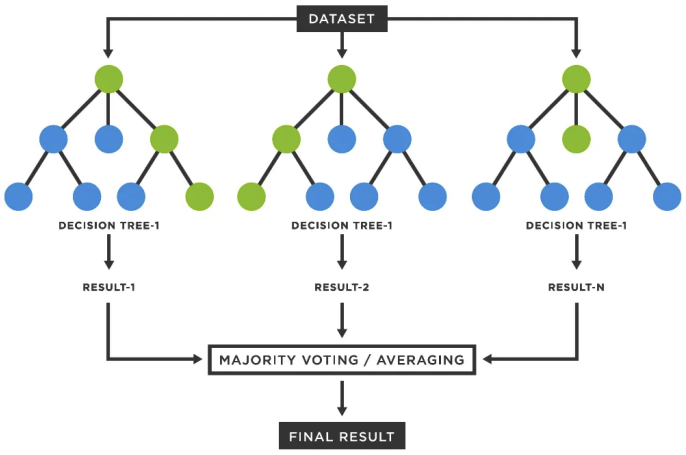

Fig. 3. Random Forest workflow showing individual decision trees and aggregation through majority voting or averaging. [9]

IV. Results

A. Evaluation Metric: AUC-ROC

In this study, the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) was selected as the primary metric for evaluating model performance. Metrics like accuracy can be misleading in imbalanced datasets, as they tend to favor the majority class. For example, a model predicting all defaulters as non-defaulters could still achieve high accuracy, despite being fundamentally flawed and potentially causing significant financial damage in real-world scenarios. AUC-ROC offers a more balanced evaluation by assessing the model's ability to distinguish between positive (default risk) and negative (no default) classes, irrespective of class proportions [12].

The AUC-ROC is computed based on two key rates: the True Positive Rate (TPR) and the False Positive Rate (FPR). The True Positive Rate, also known as sensitivity or recall, measures the proportion of actual positives that are correctly identified by the model. It is mathematically defined as [12]:

$$TPR = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{7}$$

This metric indicates the model's effectiveness in identifying positive cases, such as clients at risk of default. A higher TPR reflects better detection of actual positive instances.

On the other hand, the False Positive Rate measures the proportion of actual negatives that are incorrectly classified as positives. It is mathematically expressed as [12]:

$$FPR = \frac{\text{False Positives (FP)}}{\text{False Positives (FP)} + \text{True Negatives (TN)}} \tag{8}$$

TABLE I
MODEL PERFORMANCE COMPARISON BASED ON AUC-ROC SCORES

Model Candidates	Parameter	Average AUC-ROC Score on 5-fold Validation
Logistic Regression	`'class_weight': balanced, 'C': 0.01, 'max_iter': 100, 'penalty': 'l1', 'solver': 'saga'`	0.715516
Random Forest	`'class_weight': balanced, 'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 500`	0.782441
Linear SVM	`'class_weight': balanced, 'kernel': linear, 'C': 0.01`	0.714386
Non-linear RBF SVM	`'class_weight': balanced, 'kernel': rbf, 'C': 1, 'gamma': 0.01`	0.766761
Non-linear Polynomial SVM	`'class_weight': balanced, 'kernel': poly, 'C': 0.1, 'coef0': 1, 'degree': 3, 'gamma': scale`	0.765765

The FPR quantifies the likelihood of flagging a safe client as risky, which could lead to unnecessary interventions or resource allocation.

The AUC-ROC curve is constructed by plotting the TPR against the FPR at various classification thresholds, providing a comprehensive view of the model's performance across different decision boundaries. A higher AUC-ROC score signifies a better ability to distinguish between the positive and negative classes.

Table I summarizes the performance of various machine learning models based on their average AUC-ROC scores from 5-fold cross-validation. Among these models, the Random Forest achieved the highest performance with an AUC-ROC score of 0.782. This was followed by the SVM with either an RBF or polynomial kernel, both achieving similar AUC-ROC scores of approximately 0.76. In contrast, Logistic Regression and Linear SVM performed the worst, with AUC-ROC scores around 0.71. The table also details the best parameter configurations for each model. Based on these results, the Random Forest model was selected for further evaluation on the test dataset.

B. Evaluation of the Final Model on the Test Dataset

The Random Forest model, selected as the final model, was evaluated on the test dataset and achieved an AUC-ROC score of 0.7747, as shown in Figure 4. This demonstrates its ability to effectively differentiate between clients at default risk (positive class) and those not at risk (negative class). The ROC curve illustrates the trade-off between the True Positive Rate and False Positive Rate across various thresholds, remaining well above the diagonal line (random guessing with an AUC of 0.50).

The AUC-ROC score and the shape of the curve show that the Random Forest model is a good choice for real-world credit risk assessment. It balances accuracy and reliability, making it effective for handling imbalanced datasets and providing reliable predictions.

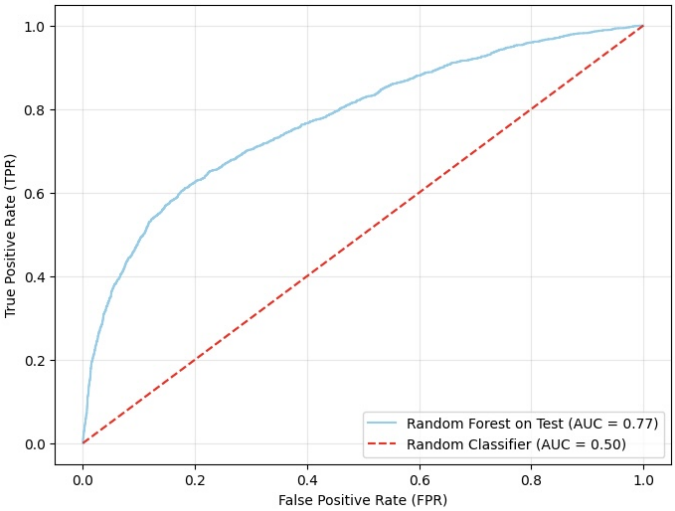

Fig. 4. Random Forest ROC Curve Test Dataset

V. CONCLUDING REMARKS

In this report, we evaluated several machine learning models for credit default prediction, including SVM, Logistic Regression, and Random Forest, with a focus on addressing the challenges of dataset imbalance. Among these, the Random Forest model emerged as the best performer, achieving an AUC-ROC score of 0.7747. This result highlights its strong ability to differentiate between default and non-default cases. The use of AUC-ROC as the primary evaluation metric ensured a balanced assessment of sensitivity (accurately identifying defaults) and specificity (minimizing false alarms), demonstrating the potential of machine learning to provide valuable insights for managing credit risk.

The Random Forest model's ability to capture complex, non-linear patterns and reduce classification errors makes it a practical and effective choice for real-world applications. Future research could focus on enhancing its performance by exploring advanced feature engineering techniques, such as incorporating temporal data or interaction effects, as well as experimenting with more sophisticated ensemble methods or deep learning approaches.

REFERENCES

[1] C. Hill, "Inclusive finance blog," https://www.experian.com/blogs/insights/inclusive-finance/, 2024.

[2] I.-C. Yeh and C.-h. Lien, "Default of credit card clients dataset," https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients, 2009, uCI Machine Learning Repository.
[3] S. Russell, "Lecture notes on supervised learning," https://people.eecs.berkeley.edu/~jrs/189/lec/04.pdf, 2024, university of California, Berkeley, CS 189 Lecture Notes.
[4] L. V. Tran, "Svm with soft margin kernel with different cases of slack variables," https://www.researchgate.net/figure/SVM-with-soft-margin-kernel-with-different-cases-of-slack-variables_fig2_327015448, 2018, accessed: December 6, 2024.
[5] E. Sahel. (2023) Introduction to rbf svm: A powerful machine learning algorithm for non-linear data. [Online]. Available: https://medium.com/@eskandar.sahel/introduction-to-rbf-svm-a-powerful-machine-learning-algorithm-for-non-linear-data-1d1cfb55a1a
[6] Sidharth. (2022, December 12) Svm kernels: Polynomial kernel - from scratch using python. [Online]. Available: https://www.pycodemates.com/2022/10/svm-kernels-polynomial-kernel.html
[7] A. W. S. (AWS), "What is logistic regression?" https://aws.amazon.com/what-is/logistic-regression/#:~:text=Logistic%20regression%20is%20a%20statistical,S%2Dcurve%20as%20shown%20below., 2024.
[8] Keylabs, "Random forest: Ensemble learning technique," https://keylabs.ai/blog/random-forest-ensemble-learning-technique/, 2024.
[9] D. Günay, "Random forest," https://medium.com/@denizgunay/random-forest-af5bde5d7e1e, 2023.
[10] S. Talazadeh and D. Peraković, "Sarf: Enhancing stock market prediction with sentiment-augmented random forest," https://arxiv.org/pdf/2410.07143, 2024.
[11] S. learn Developers, "Randomforestclassifier," https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, 2024.
[12] E. AI, "Explain roc curve," https://www.evidentlyai.com/classification-metrics/explain-roc-curve#:~:text=ROC%20AUC%20stands%20for%20Receiver,area%20under%20the%20ROC%20curve., 2024.